University of Alberta

Production-based Large Scale Construction Simulation Modeling

by

Ping Wang ©

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Construction Engineering and Management

Department of Civil and Environmental Engineering

Edmonton, Alberta

Fall 2006

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-23125-8
Our file Notre référence
ISBN: 978-0-494-23125-8

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

University of Alberta

Library Release Form

Name of Author: Ping Wang

Title of Thesis: Production-based Large Scale Construction Simulation Modeling

Degree: Doctor of Philosophy

Year this Degree Granted: 2006

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Signature

Date: Sep. 29. 2006

University of Alberta

Faculty at Graduate Studies and Research

The undersigned certify that they have read and recommended to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Production-based Large Scale Construction Simulation Modeling** submitted by **Ping Wang** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy.**

Dr. Simaan M. AbouRizk

Dr. Aminah Robinson

Dr. Dulcy Abraham

Dr. Tayfun Babadagli

Dr. Peter Flynn

Dr. Yasser Mohamed

Date: 9/19/06

ABSTRACT

An advanced modeling tool is needed in the construction industry to facilitate the implementation of promising new management theories such as lean construction and lean project delivery, and to meet project planning and control needs. Construction simulation has been used to help improve construction processes for years and is the most promising innovation of the next generation of construction management tools. The practice, however, has often been limited to modeling only subsystems, or to modeling an entire system at a very high and abstract level due to limitations in its capacity and cost-effectiveness. To meet the construction industry's needs we require the simulation of an entire construction system at the production level with a consideration of: dynamic uncertainties modeling, multiple simulation worldviews, large amounts of information, information exchange with other applications, and development by multiple developers.

In this research, the author developed the simulation-based approach to facilitate implementation of lean production in order to improve the production performance of pipe spool fabrication shops. The research was then extended to an entire industrial construction system. A special purpose large scale simulation modeling system was designed and developed for industrial construction. This system could be used to build production-based large scale simulation models. The model would provide a virtual project management laboratory, which allows construction engineers to experiment with various management strategies in planning, improving, and optimizing the entire industrial construction production system.

In current practice, developing production-based large scale construction simulation models is very challenging. In this research, these difficulties are identified through theoretical analysis and through the practical application of industrial construction simulation. The research concludes that the most effective strategy to increase the capacity and cost-effectiveness of construction simulation models is to increase knowledge standardization and reuse, model decomposability, computing ability, product representation, model openness, and model development (or data manipulation) views.

Targeting the developed strategy, the author explores several methodologies and techniques, such as High Level Architecture (HLA), Industrial Foundation Class (IFC), ontology, and eXtensible Markup Language (XML) to solve the identified challenges. A prototype architecture has been designed by integrating the proposed solutions to create increased capacity for and cost-effectiveness of production-based large scale construction simulation.

ACKNOWLEDGEMENT

First of all, my sincere thanks to my supervisor, Dr. Simaan M. AbouRizk. His advice and vision are what made this thesis possible. He has shown consistent encouragement and patience throughout my research, especially during the challenging times. His scientific and moral wisdom was indispensable in my effort towards finishing this thesis. I thank you again from my heart.

I would like to express my appreciation to KBR for sponsoring the research. I would like to thank a number of managers and engineers at KBR who shared their knowledge and experience with me. Special thanks to Mr. A.R. (Tony) Rawa, the manager of fabrication, for his strong support and collaboration.

I would like to thank my co-supervisor, Dr. Yasser Mohamed, for his help and support. He was approachable whenever I needed help. He would always share his knowledge with me. I would especially like to thank Mr. Stephen Hague who made contributions to the fundamental development of the HLA framework.

Finally, I would like to thank my parents for their continuous encouragement through all of my endeavors. My father, who is a blue-collar worker with only a junior high school diploma, encouraged me to achieve the highest academic degree possible in my childhood, though we were not sure what the highest degree in the world was at the time. Today his dream comes true.

TABLE OF CONTENTS

CHAPTER 1 – INTRODUCTION 1

- 1.1 RESEARCH MOTIVATION 1
- 1.2 CONSTRUCTION SIMULATION 5
  - 1.2.1 Background 5
  - 1.2.2 State-of-the-Art 6
- 1.3 PRODUCTION-BASED LARGE SCALE CONSTRUCTION SIMULATION 7
- 1.4 RESEARCH OBJECTIVES 8
- 1.5 RESEARCH METHODOLOGY 8
- 1.6 THESIS ORGANIZATION 9
- 1.7 REFERENCES 11

CHAPTER 2 – SIMULATION TO FACILITATE IMPLEMENTATION OF LEAN TECHNIQUE IN SPOOL FABRICATION 14

- 2.1 INTRODUCTION 14
- 2.2 INTRODUCTION TO SPOOL FABRICATION 15
- 2.3 FLOW PRODUCTION 17
  - 2.3.1 Lean Thinking 17
  - 2.3.2 Batch-and-Queue System 18
  - 2.3.3 Flow Production System 18
- 2.4 TRADITIONAL FABRICATION SHOP VS. FLOW FABRICATION SHOP 19
  - 2.4.1 Batch-and-Queue Spool Fabrication Shop 19
  - 2.4.2 Flow Spool Fabrication Shop 23
- 2.5 WEAKNESS OF VALUE STREAM MAP 26
- 2.6 SIMULATION-BASED APPROACH 27
  - 2.6.1 Choosing the Modeling Tool and Developing New Elements 27
  - 2.6.2 Simulation Model for Batch-and-Queue Fabrication Shop 32
  - 2.6.3 Simulation Model for Flow Fabrication Shop 38
  - 2.6.4 Historical Cycle Time Analysis 44
  - 2.6.5 Validation and Discussion 46
- 2.7 EXPERIMENTATION WITH THE SYSTEM USING SIMULATION 48
  - 2.7.1 Analysis of Rework Reduction 48
  - 2.7.2 Analysis of Fitting Time Reduction 50
  - 2.7.3 One-Piece-Flow Fabrication 51
  - 2.7.4 Transfer Successful Experience to Other Shops 53
- 2.8 CONCLUSIONS 54
- 2.9 REFERENCES 55

CHAPTER 3 – A LARGE SCALE SIMULATION MODELING SYSTEM FOR INDUSTRIAL CONSTRUCTION 57

- 3.1 INTRODUCTION 57
- 3.2 BACKGROUND OF INDUSTRIAL CONSTRUCTION 59
  - 3.2.1 Drafting 61
  - 3.2.2 Material Procurement 62
  - 3.2.3 Spool Fabrication 62
  - 3.2.4 Module Assembly 63
  - 3.2.5 Site Installation 65

3.3 MODELING SYSTEM DEVELOPMENT 65
  3.3.1 Product and Information Modeling 66
  3.3.2 Production System Modeling 71
  3.3.3 Typical Model Development and Features 78
3.4 CASE STUDY 83
  3.4.1 Case Description 83
  3.4.2 Model Development 86
  3.4.3 Impacts of Drawing Revisions 93
  3.4.4 Effects of Different Material Delivery Strategies 96
  3.4.5 Increase Flexibility to Accommodate Downstream Uncertainty 99
  3.4.6 Other Potential Experiments and Improvements 104
3.5 CONCLUSIONS 105
3.6 REFERENCES 107

CHAPTER 4 – PHASE ONE RESEARCH CONTRIBUTIONS AND FINDINGS 109

4.1 INTRODUCTION 109
4.2 DISCUSSION ON MODEL PERFORMANCE 110
4.3 A VIRTUAL PROJECT MANAGEMENT LABORATORY 113
4.4 IDENTIFIED CHALLENGES 115
  4.4.1 Lack of Domain Knowledge Capture, Standard, and Reuse 116
  4.4.2 Lack of Model Decomposability 117
  4.4.3 Limited Computing Ability 119
  4.4.4 Lack of Product Presentation 119
  4.4.5 Difficulties of Information Exchange with Other Applications 121
  4.4.6 Roots in AS/PI Discrete Event Simulation 122
  4.4.7 One View Development and Data Manipulation 123
4.5 CONCLUSIONS 124
4.6 REFERENCES 126

CHAPTER 5 – SOLUTIONS FOR IDENTIFIED CHALLENGES BY EXPLORING HLA, IFC, ONTOLOGY, AND XML 127

5.1 INTRODUCTION 127
5.2 INTRODUCTION TO HLA 127
  5.2.1 Origin of the HLA 127
  5.2.2 Three Components of the HLA 129
  5.2.3 How the HLA Works 133
5.3 INTRODUCTION TO IFC 133
  5.3.1 What is the IFC 133
  5.3.2 IFC Schema 134
  5.3.3 ifcXML 135
5.4 INTRODUCTION TO ONTOLOGY 136
  5.4.1 What is Ontology 136
  5.4.2 Ontology Development for Construction Industry 137
5.5 INTRODUCTION TO XML 138
  5.5.1 XML and Its Emergence 138
  5.5.2 Benefit of XML to Simulation 140
5.6 SOLUTIONS FOR THE IDENTIFIED CHALLENGES 141
  5.6.1 Domain Knowledge Acquisition, Standardization and Reuse 141
  5.6.2 Model Decomposability 142

5.6.3 Computing Ability 144
5.6.4 Product Model 144
5.6.5 Information Exchange with Other Applications 146
5.6.6 Multiple Simulation World Views 147
5.6.7 Multi-view Development / Data Manipulation 148
5.7 SUMMARY 149
5.8 REFERENCES 151

CHAPTER 6 – PROTOTYPE INTEGRATED ARCHITECTURE 154

6.1 INTRODUCTION 154
6.2 LEVEL 1: KNOWLEDGE STRUCTURE AND ACQUISITION 157
6.2.1 Knowledge to Acquire 157
6.2.2 Knowledge Structure 160
6.2.3 Three Ways of Knowledge Acquisition 163
6.2.4 Knowledge Tailoring 167
6.3 LEVEL 2: STRUCTURED KNOWLEDGE LIBRARY 167
6.3.1 Extensible Object Model Library 167
6.3.2 Process Model Library 169
6.3.3 Domain-Independent Model Library 170
6.3.4 Advantages of Knowledge Standardization and Library 170
6.4 LEVEL 3: BUILDING A MODEL 171
6.4.1 Modeling Elements 172
6.4.2 Generate Specific eFOM 172
6.4.3 HLA-compliant Simulation Model Development 173
6.5 LEVEL 4: EXECUTION 174
6.5.1 Simulation Services 175
6.5.2 Execution on Multiple Computers 176
6.6 SIDE LEVEL: MULTI-VIEW DEVELOPMENT / DATA MANIPULATION 176
6.6.1 Increased Efficiency of Scenario Iterations 179
6.6.2 Knowledge Acquisition from Other Standards 181
6.6.3 Data Exchange with Other Applications 181
6.6.4 Output Utilization and Display 182
6.7 CASE STUDY 183
6.7.1 Knowledge Acquisition, Standardization, and Library 183
6.7.2 Generate a Specific eFOM 186
6.7.3 Design and Develop a Simulation Federation 186
6.7.4 Demonstration of the Developed Federation 193
6.8 CONCLUSIONS 194
6.9 REFERENCES 196

CHAPTER 7 – FINAL DISCUSSION 197

7.1 RESEARCH SUMMARY 197
7.2 SUMMARY OF RESEARCH CONTRIBUTIONS 198
7.2.1 Academic Contributions 198
7.2.2 Contributions to the Construction Industry 199
7.3 RECOMMENDATIONS FOR FUTURE RESEARCH 201
7.3.1 Detail Modeling System of Site Installation 201
7.3.2 Enrich the Knowledge Library 202
7.3.3 Objectify Standard Processes 202
7.3.4 A Standard Data Model in XML Format Simulation Model 203
7.4 REFERENCES 204

APPENDIX 1 – INDUSTRIAL CONSTRUCTION SIMULATION MODELING SYSTEM USER'S GUIDE 205

1. OVERVIEW OF MODELING SYSTEM 205
2. PRODUCT AND INFORMATION MODELING SYSTEM 205
3. PRODUCTION MODELING SYSTEM 206

APPENDIX 2 – DEVELOPMENT CODE OF SIMULATION MODELING SYSTEM FOR INDUSTRIAL CONSTRUCTION 233

1. DRAFTING TEMPLATE 233
2. MATERIAL PROCUREMENT TEMPLATE 239
3. SPOOL FABRICATION TEMPLATE 241
4. MODULE ASSEMBLY TEPLATE 266
5. SITE INSTALLATION TEMPLATE 269
6. PUBLIC MODULE ELEMENTS TEMPLATE 270

APPENDIX 3 – A COMPLETE CSOM FOR THE DOMAIN OF INDUSTRIAL CONSTRUCTION 292

1. XML FILE OF THE CSOM 292

TABLE OF TABLES

Table 2-1. Selected Modeling Elements of Common Template in Simphony 28
Table 2-2. New Elements Enhancing Common Template of Simphony.. 31
Table 2-3. Data Collection Summary for Modeling Old System 35
Table 2-4. Data Collection Summary for Modeling New System... 41
Table 3-1. SPS Modeling Elements for Industrial Construction 72
Table 3-2. Public Module Elements for Industrial Construction... 75
Table 3-3. Properties of Entities 85
Table 3-4. Drawing Revision Analysis... 94
Table 6-1. Relationships between Challenges and Methodologies/Techniques........................... 154

TABLE OF FIGURES

Figure 1-1. Research Methodology Chart 9
Figure 2-1. Spool Fabrication Process 16
Figure 2-2. Layout of the Old Shop 20
Figure 2-3. Value Stream Map of Old System 22
Figure 2-4. Layout of the New Shop 24
Figure 2-5. Value Stream Map of the New System 25
Figure 2-6. Simulation Model of Batch-and-Queue Fabrication Shop 33
Figure 2-7. Detailed Process Model of a Roll Fitting Working Station and a Roll Welding Working Station 34
Figure 2-8. Cycle Time Distribution of Simulation Output of Batch-and-Queue Fabrication Shop 38
Figure 2-9. Simulation Model of Flow Fabrication System 39
Figure 2-10. Detailed Process Modeling of a Work Cell 40
Figure 2-11. Cycle Time Distribution of Simulation Output of Flow Fabrication System 44
Figure 2-12. Cycle Time Distribution of Batch-and-Queue Fabrication System from Historical Data 45
Figure 2-13. Cycle Time Distribution of Flow Fabrication System from Historical Data 46
Figure 2-14. Modeling Rework Reduction 49
Figure 2-15. Sensitivity Analysis of Average Cycle Time to Rework Reduction 50
Figure 2-16. Sensitivity Analysis of Average Cycle Time to Fitting Time Reduction 51
Figure 2-17. Rebuilt Model of One-Piece-Flow Production of a Work Cell 52
Figure 2-18. Cycle Time Distribution of Simulation Output of One-Piece-Flow Fabrication System 53
Figure 3-1. Typical Production Process of Industrial Construction 61
Figure 3-2. Module Assembly Process 64
Figure 3-3. Industrial Construction Modeling System Architecture 66
Figure 3-4. Work Breakdown Structure 67
Figure 3-5. Sample User Interface of Control System for Product Modeling 70
Figure 3-6. Sample User Interface of Control System for Production System 78
Figure 3-7. Modeling Flow Diagram (a) 79
Figure 3-8. Modeling Flow Diagram (b) 80
Figure 3-9. Modeling Flow Diagram (c) 81
Figure 3-10. Modeling Flow Diagram (d) 82
Figure 3-11. Project of Case Study 84
Figure 3-12. First Level of the Developed Model 87
Figure 3-13. Display of Hierarchical Model of Drafting System 88
Figure 3-14. Display of Hierarchical Model of Material Procurement 89
Figure 3-15. Display of Hierarchical Model of Fabrication 90
Figure 3-16. Display of Hierarchical Model of Module Assembly 91
Figure 3-17. Display of Hierarchical Model of Site Installation 92
Figure 3-18. Modeling Drawing Revision 95
Figure 3-19. Sensitivity Analysis of Project Duration to Drawing Revision 96
Figure 3-20. Modeling Material Delivery Strategy 1 97
Figure 3-21. Modeling Material Delivery Strategy 2 98
Figure 3-22. Modeling Material Delivery Strategy 3 98
Figure 3-23. Analysis of Project Duration to Different Material Delivery Strategies 99
Figure 3-24. Illustration of the Modeling Logics 100
Figure 3-25. Traced Travel Information of Spools 101

Figure 3-26. Original Spool Fabrication Schedule 102
Figure 3-27. Updated Spool Fabrication Schedule 103
Figure 4-1. Relationship Between Fabrication Time and Total Diameter Inch of Spools 111
Figure 4-2. Foundation to Increase Capacity and Cost-effectiveness of Production-based Large Scale Construction Simulation Models 125
Figure 5-1. Logical View of A HAL Federation 133
Figure 5-2. Schema Overview of IFC2x Edition 2 (Liebich 2004) 135
Figure 5-3. Acquire Standard Knowledge from IFC and Developed Construction Ontology 142
Figure 5-4. A Large Scale Model is Decomposed to Component Models 143
Figure 5-5. Product Information Generation 145
Figure 5-6. Multi-view Development / Data Manipulation of a Simulation Model 149
Figure 6-1. Architecture of Production-based Large Scale Construction Simulation Modeling. 156
Figure 6-2. Core Layer of CSOM 161
Figure 6-3. Domain-specific Layer of CSOM 163
Figure 6-4. Import Objects Model from ifcXML to CSOM 164
Figure 6-5. Import Object Model from the Developed Construction Ontology to CSOM 165
Figure 6-6. Model Experts' Knowledge Using UML to Generate CSOM 166
Figure 6-7. Model Experts' Knowledge Using XML Editor to Generate CSOM 167
Figure 6-8. Sample CSOM for the Domain of Industrial Construction 169
Figure 6-9. Generate eFOM from CSOM 173
Figure 6-10. Build Simulation Model Using Model Library 174
Figure 6-11. Execution on Multiple Computers 176
Figure 6-12. XML Text File of a Simulation Model 178
Figure 6-13. View and Manipulate a XML-based Simulation Model 179
Figure 6-14. Resource Control Panel based on the XML Model File 180
Figure 6-15. Knowledge Acquisition 181
Figure 6-16. Data Exchange between Simulation Models and Other Applications 182
Figure 6-17. UML Static Model for Industrial Construction CSOM 184
Figure 6-18. A Complete CSOM for the Industrial Construction Project 185
Figure 6-19. Flow Diagram of Drafting Federate 189
Figure 6-20. Flow Diagram of Procurement Federate 190
Figure 6-21. Flow Diagram of Fabrication Federate 191
Figure 6-22. Flow Diagram of ModuleAssembly Federate 192
Figure 6-23. Flow Chart of Federation "IndustrialConstruction" 193
Figure 6-24. Demonstration of Execution of the Federation 194

CHAPTER 1 – INTRODUCTION

1.1 RESEARCH MOTIVATION

Why do we need to develop production-based large scale construction simulation models? Are there efficient modeling tools and modeling approaches currently available to build these models? The following discusses several driving forces for this research.

1. Project Management vs. Production Management

There are several techniques for planning and managing construction projects. Most of them are based on the project management (PM) framework. The critical path method (CPM) is the most popularly used underlying method of these techniques. Problematically, CPM assumes that activity durations are constant, which is not true in construction projects. The Program Evaluation and Review Technique (PERT) is similar to CPM, the only difference being that activity duration is represented with Beta distributions to model the uncertainty. Monte-Carlo simulation is a probabilistic approach allowing for the representation of the element of risk in project plans.

Flaws have been found in the PM framework in the context of managing construction projects. The PM framework is derived from the activity-centered approach found in mass production and project management. The aim is to optimize the project activity-by-activity (Howell 1999). The extent, to which the above-described planning methods work, however, depends on the availability of the required resources for activities as they are needed. Some resource allocation methods are used to "level" resources in order to obtain a revised schedule (Russel and Dubey 1995). They still fail, however, to represent complex logic, dynamic interactions between resources and processes, and different construction methods utilized. The PM framework-based construction management methodology does not provide the theory for describing the mechanics of a construction production system. Project control under the PM framework focuses on identifying variances between the project plan and the actual results, and on adjusting resources to meet the master schedule.

A construction system is not an activity-centered system. It actually produces products using resources. Construction researchers have been looking for new theories for construction management. Some researchers explored techniques and methodologies in production management theory. "Lean Construction" terminology was coined in the 1990s, and was the most important of the new emerging theories. Lean construction is derived from lean production, which originated in the manufacturing industry. Principles and techniques of lean theory have been transferred into the domain of construction management for some time. Managing construction using lean concepts relies on a production management worldview. Therefore, only when the production problems of a construction system are identified and understood, can new opportunities be seen. Under lean theory, project control should primarily refer to causing a desired future rather than merely identifying variances between the planned and the actual (Ballard 2000).

2. Construction Management Modeling Tools

As conveyors of PM methodology, PM-based modeling tools are most often used. Some computer systems, such as Primavera Project Planner (P3) and Microsoft Project, implement the CPM method and the bar chart to simplify constructing and managing plans. These systems also provide functions for representing projects hierarchically and levelling resources. These modeling tools, however, cannot capture the mechanics of a construction production system.

Construction simulation has been used to help improve construction processes for years. Several construction simulation tools, such as CYCLONE (CYClic Operation NEtwork), STROBOSCOPE, and *Simphony* have been developed. The simulation-based modeling technique is considered a powerful tool for modeling construction operations. It can describe complex system logic and the dynamic interactions between resources and processes. It is capable of capturing the production mechanics of a construction system. It can be used to test different construction methods to do experimental planning for a given project. Simulation, however, has not been widely used in the construction industry as a daily operational tool due to its limited capacity, limited cost-effectiveness, and the expertise needed for operation.

3. Simulate an Entire Construction System at the Production Level

The research successes of construction simulation achieved thus far have been limited to modeling an entire system at a very high abstract level, or to modeling the construction processes of a subsystem. The former fails to capture the production mechanics of the construction system. The latter is incapable of properly or adequately answering questions beyond the scope of local processes; a large and complex construction system cannot optimally improve the overall performance without an understanding of how the processes of subsystems affect the larger system.

A suitably detailed simulation model covering an entire construction system behaves differently. It can capture and reflect product features, production processes, resource allocations and interactions, activity interactions, dependence, variation, and the impact of site conditions at the production level. Only when a system is modeled at the production level can construction engineers experiment with different scenarios and test all types of uncertainties at that level. This kind of simulation model can also capture the interactions of subsystems and thereby identify problems among subsystems. It can work to improve the entire system's performance.

Lean construction theory is another driving force behind modelling the entire construction system at the production level. This type of simulation model enables testing a variety of production-derived lean techniques; and it allows construction engineers to experiment with different strategies of the Lean Project Delivery System (LPDS) developed by The Lean Construction Institute (LCI) to improve the whole project delivery and supply chain (Ballard 2000).

4. Increase Capacity and Cost-effectiveness of Simulation Model

It is very time and effort consuming to build the desired construction simulation models using currently available tools, other issues are implied besides the model scale. In order to model the construction system realistically, one must consider implementing dynamic modeling of uncertainties using advanced methods within the simulation model. The model might be required to have multiple simulation world-views interacting, such as continuous simulation, state-based simulation, and subjective simulation other than discrete-event simulation. These models will be driven by a large volume of information; as such, they will need to be open models capable of exchanging information with other applications in the construction industry. At some point the models may need to be developed by multiple developers in order to meet the simulation project lead-time. The development of such models presents many challenges, which will be discussed in detail in Chapter 4. In addition, captured knowledge and developed component models cannot easily be reused in future developments. It is necessary, therefore, to explore theoretical solutions in order to increase the capacity of construction simulation models and to improve the cost-effectiveness of their development. The solutions discovered need to be incorporated into a simulation tool or platform in order to enhance model development and application.

1.2 CONSTRUCTION SIMULATION

1.2.1 Background

There have been more than ten construction simulation tools developed in construction research since Halpin (1977) developed the first construction simulation system: CYCLONE (CYClic Operation NEtwork). With CYCLONE, users built models using a provided set of abstract but simple constructs. CYCLONE popularized the use of simulation in construction research.

CYCLONE subsequently fostered a wide range of construction simulation research efforts. These include: MicroCYCLONE (Lluch and Halpin 1981), INSIGHT (Paulson 1987), RESQUE (Chang and Carr 1987), UM-CYCLONE (Ioannou 1989), COOPS (Liu and Ioannou 1992, 1993), CIPROS (Odeh et al. 1992), STEPS (McCahill and Bernold 1993), STROBOSCOPE (Martinez and Ioannou 1994), DISCO (Huang et. al. 1994), and ABC (Shi 1999). All these tools were derived from CYCLONE, with the intention of enhancing its functionality. The underlying simulation strategy of these systems is activity scanning (AS). Certain approaches, such as COOPS and STROBOSCOPE, focus on enhancing resource presentation, enabling resource attribute definition.

Chang (1991) introduced object-oriented concepts to construction simulation modeling. Object-oriented simulation models resemble their real life counterparts. The use of the object-oriented approach leads to reduced coding and improved simulation model readability (Oloufa 1993).

Tommelein (1994), Oloufa (1994) and Shi (1997) implemented a library-based modeling approach. Project simulation models were assembled from a set of pre-defined components. This approach enabled construction engineers to take advantage of simulation without much initial simulation knowledge.

Sawhney (1996) used modular concepts as defined by Ziegler (1984) to develop hierarchical simulation modeling (HSM) systems for modelling construction projects. The basic process modeling of HSM used the CYCLONE methodology.

1.2.2 State-of-the-Art

Special purpose simulation (SPS) tools were later introduced to the construction domain. These simulation modeling tools focus on one particular domain of construction operations and facilitate modeling projects within that domain. In 1999, *Simphony*, a unified construction simulation modeling environment for building SPS tools, was developed by Hajjar and AbouRizk (1999) to reduce the time required to develop SPS tools.

Simphony is an open modeling platform. The simulation strategy of *Simphony* at the base level is event scheduling (ES). The simulation strategy of the developed tool (e.g. Common Template, CYCLONE) implemented on this platform can be either activity scanning (AS) or process interaction (PI). Monte Carlo simulation-based tools can also be developed in *Simphony*. In theory all construction simulation tools mentioned in Section 1.2.1 can be redeveloped in the *Simphony* environment with reduced time and effort.

1.3 PRODUCTION-BASED LARGE SCALE CONSTRUCTION SIMULATION

Production-based large scale construction simulation models are defined based on the above discussions. This type of simulation model is a project scope simulation with production level details and complex logic. This model covers the drafting, procurement, fabrication, and construction phases, and encompasses multiple time scales (i.e., component models in a large system may have different time units). Instead of employing only PI or AS discrete-event simulation, the model might have multiple simulation world-views interacting such as: continuous simulation, state-based simulation, and subjective simulation (Fishwick 1995). The model is driven by a large amount of product and site condition information. It is an open model capable of exchanging information with other applications. It would need to be developed by multiple developers when necessary in order to meet simulation project lead-time.

Production-based large scale construction simulation modeling is not a completely new concept. Research has been conducted on several of its features. However, this particular concept is a new trend for construction modeling and simulation: a production-based large scale construction simulation model at the project level behaves differently and can provide more "what-if" proactive tests to construction engineers. The aim of this research is to promote the application of construction simulation by improving its capabilities and by increasing the efficiency of its development.

1.4 RESEARCH OBJECTIVES

The objectives of this research are summarized as follows:

1. To demonstrate and prove the necessity and contributions of production-based large scale construction simulation.
2. To identify challenges in the development of production-based large scale construction simulation using current construction simulation tools.
3. To discover solutions to the identified challenges and to integrate these solutions into one development environment in order to achieve increased capacity and cost-effectiveness of production-based large scale construction simulation.

1.5 RESEARCH METHODOLOGY

In order to accomplish the above objectives, the following methodologies will be adopted. Figure 1-1 illustrates structure of these research methodologies.

1. Review the available literature on construction project planning techniques, construction production management theory, construction simulation, and all construction simulation tools.
2. Develop simulation models for spool fabrication shops to facilitate implementation of lean production.
3. Develop a large scale simulation modeling system for industrial construction. Build of models using the developed system, test and validate the models, and experiment with different scenarios.

4. Identify challenges surrounding the development of production-based large scale construction simulation models through theoretical analysis and the development process.
5. Propose solutions to identified challenges by exploring new theories, methodologies, and techniques.
6. Design an integrated architecture by combining the above proposed solutions.
7. Implement part of the architecture based on the *Simphony* environment and develop a pilot case model of industrial construction simulation to demonstrate its feasibility.

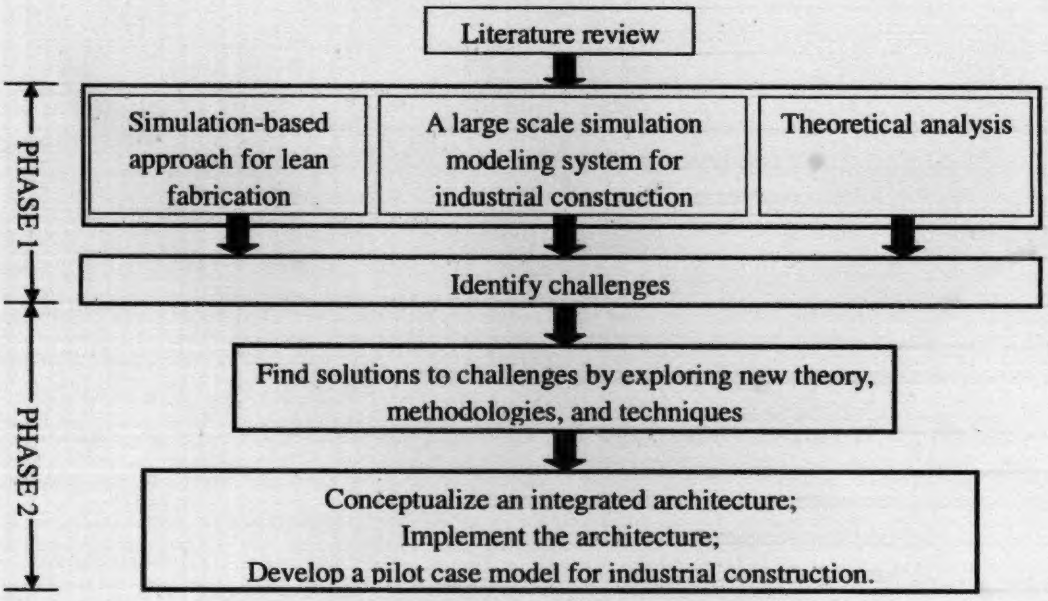

Literature review
Simulation-based approach for lean fabrication
A large scale simulation modeling system for industrial construction
Theoretical analysis
Identify challenges
Find solutions to challenges by exploring new theory, methodologies, and techniques
Conceptualize an integrated architecture;
Implement the architecture;
Develop a pilot case model for industrial construction.
PHASE 1
PHASE 2

Figure 1-1. Research Methodology Chart

1.6 THESIS ORGANIZATION

- Chapter 2 discusses an application of simulation in industrial construction fabrication that facilitates the implementation of lean production principles.

- Chapter 3 describes the development of a large scale simulation modeling system tailored for industrial construction and its application.
- Chapter 4 summarizes the contributions of the first phase of research and presents a list of identified challenges.
- Chapter 5 introduces new theories, methodologies, and techniques, and proposes solutions for the identified challenges.
- Chapter 6 describes a prototype integrated architecture of production-based large scale construction simulation. Part of the proposed integrated architecture is implemented. A case study for industrial construction is conducted.
- Chapter 7 provides the final discussion and future research recommendations.

1.7 REFERENCES

Ballard, G. (2000). *Lean Project Delivery System*, Lean Construction Institute.

Chang, D. Y., and Carr, R. I. (1987). "RESQUE: A Resource Oriented Simulation System for Multiple Resource Constrained Processes." *Proceedings of the 1987 PMI Seminar/Symposium*, Milwaukee, Wisconsin, 4-19.

Chang, D. Y. (1991). "Object-Oriented Simulation System for Construction Process Planning." *Construction Congress 91*, ASCE, New York, NY, USA, 626-631.

Hajjar, D., and AbouRizk, S. (1999). "Simphony: An Environment for Building Special Purpose Construction Simulation Tools." *Proceedings of the 1999 Winter Simulation Conference*, ed. Phillip A. Farrington, Harriet Black Nembhard, David T. Sturrock, and Gerald W. Evans. Phoenix, Arizona, USA, 998-1006.

Hajjar, D., and AbouRizk, S. (2002). "Unified modeling methodology for construction simulation", *Journal of Construction Engineering and Management*, 128 (2) 174-185.

Halpin, D. W. (1977). "CYCLONE: Method for Modeling of Job Site Processes" *Journal of the Construction Division*, ASCE, 103(3), 489-499.

Howell, G. A. (1999). "What is Lean Construction." *Proceedings Seventh Annual Conference of the International Group for Lean Construction*, IGLC-7, Berkeley, CA, 1-10.

Huang, R., Grigoriadis, A. M., and Halpin, D. W. (1994). "Simulation of Cable-stayed Bridges Using DISCO." *Proceedings of Winter Simulation Conference*, ASCE, 1130-1136.

Ioannou, P. G. (1989). *UM-CYCLONE User's Guide*, Department of Civil Engineering, The University of Michigan, Ann Arbor, Michigan.

Johnston, D. W. (1981). "Linear Scheduling Method for Project Planning Analysis." *Journal of Construction Engineering and Management*. ASCE. 107(2), 247-261.

Liu, L. Y., and Ioannou, P. G. (1992). "Graphical Object-Oriented Discrete-Event Simulation System." *Proceedings of Winter Simulation Conference*, ASCE, 1285-1291.

Liu, L. Y., and Ioannou, P. G. (1993). "Graphical resource-based object-oriented simulation for construction process planning." *Proceedings of the 5th International Conference on Computing in Civil and Building Engineering - V-ICCCBE*. Anaheim, CA, USA

Lluch, J. F., and Halpin, D. W. (1981). "Analysis of Construction Operations Using Microcomputers." *Journal of the Construction Division*, ASCE Vol. 108 No. C01:129-145.

Martinez, J., and Ioannou, P. G. (1994). "General Purpose Simulation with Stroboscope." *Proceedings of Winter Simulation Conference*, ASCE, 1159-1166.

McCahill, D. F., and Bernold, L. E. (1993) "Resource-oriented modeling and simulation in construction.'' *Journal of Construction Engineering and Management*, ASCE, 119(3), 590–606.

Odeh A. M., Tommelein I. D., and Carr R. I. (1992) "Knowledge-Based Simulation of Construction Plans." *Proceedings of the Eighth Conference on Computing in Civil Engineering*, ASCE, Dallas, Texas. 1042-1049.

Oloufa, A. A. (1993). "Modeling Operational Activities in Object-Oriented Simulation." *Journal of Computing in Civil Engineering*, ASCE, 7(1), 94-106.

Oloufa, A. A. (1994). "User-oriented approach to construction simulation of buildings." *Microcomputers-in-Civil-Engineering*, 9(6), 1994, 425-433.

Paulson, B. C. Jr., Chan, W. T., Koo, C. C. (1987). "Construction Operation Simulation by Microcomputer." *Journal of Construction Engineering and Management*, ASCE, 113(2), 302-314.

Russell, A., and Dubey, A. (1995). "Resource Leveling and Linear Scheduling." *Proceedings of the Second Congress* held in conjunction with A/E/C Systems held in Atlanta, Georgia, June 5-8, 1995.

Sawhney, A., and AbouRizk, S. M. (1996). "Computerized Tool for Hierarchical Simulation Modeling." *Journal of Computing in Civil Engineering*, 10(2), 115-124.

Shi, J. (1999). "Activity-based construction (ABC) modeling and simulation method." *Journal of Construction Engineering and Management*, ASCE. 125(5), 354-360.

Shi, J., and AbouRizk, S. M. (1997). "Resource-Based Modeling for Construction Simulation." *Journal of Construction Engineering and Management*, 123(1), 26-33.

Tommelein, I. D., Carr, R. I., Odeh, A. M. (1994). "Knowledge-Based Assembly of Simulation Networks using Construction Designs, Plans, and Methods." *Proceedings of the 1994 Winter Simulation Conference*. Buena Vista, FL, USA.

Zeigler, B. P. (1984). *Multifaceted Modeling and Discrete Event Simulation*. Academic Press, London and Orlando, Fla.

CHAPTER 2 – SIMULATION TO FACILITATE IMPLEMENTATION OF LEAN TECHNIQUE IN SPOOL FABRICATION

2.1 INTRODUCTION

The fabrication and construction phase of an industrial construction project typically involves drafting, shop fabrication, module assembly, and site installation. The proportion of shop fabrication in an industrial construction project has been on the rise due to better production management in the controllable shop environment. Reducing fabrication cycle time and its variations can reduce fabrication costs, expedite progress, benefit the scheduling of module assembly and site construction, and, ultimately, improve the entire project delivery. However, industry practice indicates that the spool fabrication shop in particular is inefficiently managed due to its construction characteristics of product uniqueness, lack of sufficient automation, labor-intensiveness, and due to the frequent change orders issued by the client during production.

These characteristics make the application of new production management theory and techniques very challenging. Lean theory has been widely applied in the manufacturing industry in the last few decades, and has proved to be very beneficial. In the construction industry, the Lean Construction Institute (LCI) has begun work to disseminate the concepts of "lean delivery". Unfortunately, academically-developed principles have yet been accepted widely in the construction industry. For example, lean techniques are rarely used by spool fabricators.

Compared to manufacturing, spool fabrication lacks a suitable tool for modeling and analyzing system changes and improvements. The value stream map (VSM) is a popular tool that can facilitate the application of lean production, but does not yet represent the dynamic nature and wide uncertainty of a spool fabrication shop in an efficient manner. Simulation has been widely used and sometimes is the only appropriate tool for production system analysis; however, its use in a spool fabrication shop is challenging due to the above-mentioned characteristics.

This research is based on the production practice of an Edmonton-based industrial construction contractor that is applying lean techniques in its spool fabrication shops. This chapter initially introduces the background of industrial construction fabrication and its existing problems. Next, flow production, one of the lean principles, is explained and discussed. The two systems of a traditional fabrication shop and a flow fabrication shop are compared in Section 2.4. The weakness of the value stream map is discussed in Section 2.5. Section 2.6 presents two developed simulation models for the two systems. Historical cycle time is extracted and analyzed to compare the simulation results. The inner features and differences between the two systems are revealed by comparisons. In Section 2.7 more experiments are done using the developed models to demonstrate more potential improvements for the new system. Conclusions are drawn in Section 2.8.

2.2 INTRODUCTION TO SPOOL FABRICATION

In order to achieve higher productivity and quality, much of the work in industrial construction has been moved to the shop. The typical operations of pipe spool fabrication include: cutting, fitting, welding, quality control (QC) checking, stress relief, hydro testing, painting, and other surface finishing. A spool is normally decomposed into pipes and fittings (elbow and flange). During the cutting stage, raw pipes are cut to the required sizes. They are then fitted and welded. Welded spools are tested by the quality control crew. After quality control checking, a spool may need to undergo any or all of the following operations: stress relief, hydro testing, painting, and other surface finishing. These typical processes are shown in Figure 2-1. Spool fabrication has become a critical stage for the whole project delivery in industrial construction. Its performance directly influences the downstream stages, module assembly, and site construction.

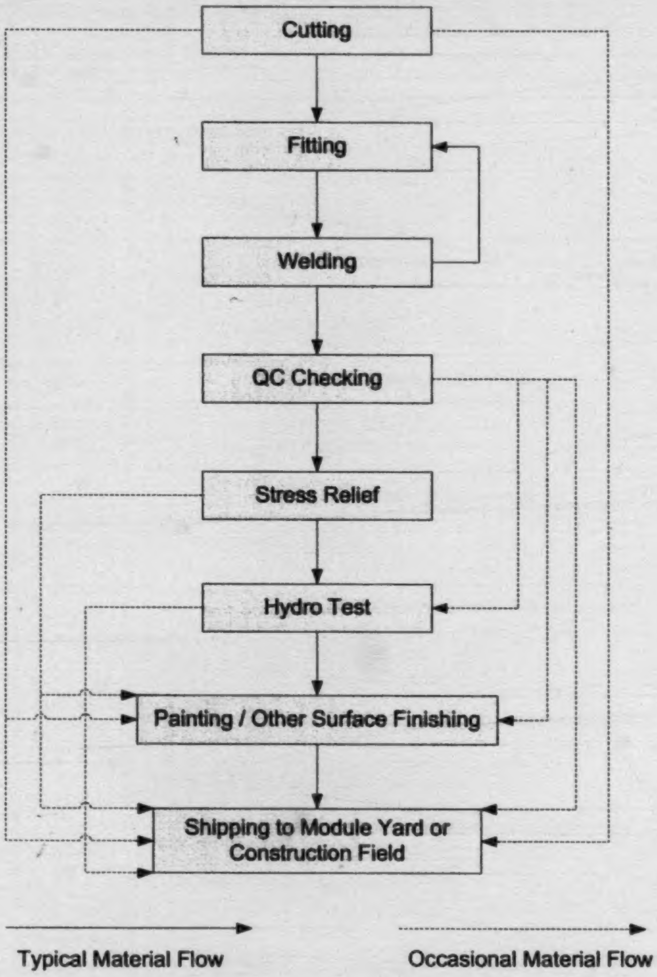

Cutting
Fitting
Welding
QC Checking
Stress Relief
Hydro Test
Painting / Other Surface Finishing
Shipping to Module Yard or Construction Field
Typical Material Flow
Occasional Material Flow

Figure 2-1. Spool Fabrication Process

A spool fabrication shop appears to be much the same as a manufacturing shop; however, every product is unique and the product family is extremely various. The process is also labor-intensive, less automated, and interrupted by frequent change orders issued by clients in the midst of the fabrication. These characteristics make daily shop management difficult. It also presents a challenge to production scheduling. Scheduling systems for pure manufacturing flow cannot be easily used in spool fabrication shops due to the many uncertainties of the fabrication process. PM-base modeling tool such as Microsoft Project and P3 (Primavera Project Planner) cannot be well applied effectively because such a fabrication shop processes numerous unique products, rather than being activity-oriented. They also make the application of new production management techniques, such as lean techniques, very challenging.

2.3 FLOW PRODUCTION

2.3.1 Lean Thinking

Lean thinking has proven very beneficial in improving production process and product quality over the last few decades. Lean production techniques have been widely applied in the manufacturing industry. Lean thinking is all about the elimination of waste. In lean theory, waste can be defined as over or under production, wait time, transportation, inappropriate processing, unnecessary materials inventory, unnecessary motion, and product defects. Under the lean thinking umbrella there are principles, methods, techniques, and tools to facilitate its implementation. They are applied to eliminate one or more of the above-defined wastes. Lean thinking has five basic principles: (1) know the real value, (2) map the value stream, (3) flow, (4) pull, and (5) perfection (Womack, and Jones 1996). This research focuses on the investigation and implementation of one of the lean principles: flow production.

2.3.2 Batch-and-Queue System

The common practice of many manufacturing systems a few decades ago was the batch-and-queue system. For example, Womack and Jones (1996) investigated and analyzed the bicycle industry, which was a highly disintegrated traditional batch-and-queue system and differentiated production activities by type. Many industries even designed and built departments for each type of activity such as cutting, bending, fitting, welding, or painting. Materials or parts were processed batch by batch for each activity, and then were sent to inventory to await the next activity. When the products are highly mixed, the changeover of tools is frequent and wastes time. In order to reduce the frequency of changeover, larger batches are processed. This gives way, however, to the problem of tracking inventory and of sending parts to the next correct process at the right time. Such a system caused several types of waste as defined by lean thinking: overproduction, wait time, transportation, inventory, and motion. Furthermore, the total lead time of the batch-and-queue system is lengthy due mainly to the above wastes.

2.3.3 Flow Production System

Flow, one of the five basic principles of lean thinking was described by Womack and Jones (1996) as organizing "all of the essential steps needed to get a job done into a steady, continuous flow, with no wasted motions, no interruptions, no batches, and no queues", and that the end objective of flow is "to totally eliminate all stoppages in an entire production process". In a flow system, the operation activities are arranged in a sequence, often arranged in a "U"-shaped cell. If only one product moves at a time, it is called "one-piece-flow".

Flow systems greatly reduce the waste caused by a batch-and-queue system. They reduce the wait time and delays occurring in batching processing. They also reduce inventory and transportation, which subsequently results in less laborer motion and space occupation. Implementing a flow system, however, especially a one-piece-flow method, has certain requirements: the changeover time of a machine should be short or even instant from one product specification to the next; the size of a product should be suitable because too small a product is unsuitable when setup time cannot be overlooked; and laborers need to be cross-trained to become able to do more tasks.

2.4 TRADITIONAL FABRICATION SHOP vs. FLOW FABRICATION SHOP

2.4.1 Batch-and-Queue Spool Fabrication Shop

The Edmonton-based industrial construction contractor in this research had used the batch-and-queue fabrication system for many years. They have five shops, each of which have the same process with comparable facilities, equipment, and laborers. They fabricate spools of different sizes and materials. By designing five similar production shops to fabricate different product families, instead of assigning one shop for each activity, the company unintentionally involved lean thinking. Each shop, however, was still using a batch-and-queue system. Figure 2-2 depicts the layout of one of the five shops, which has one cutter, three roll-fitters, five roll-welders, one position-fitter, one position-welder, and six cranes.

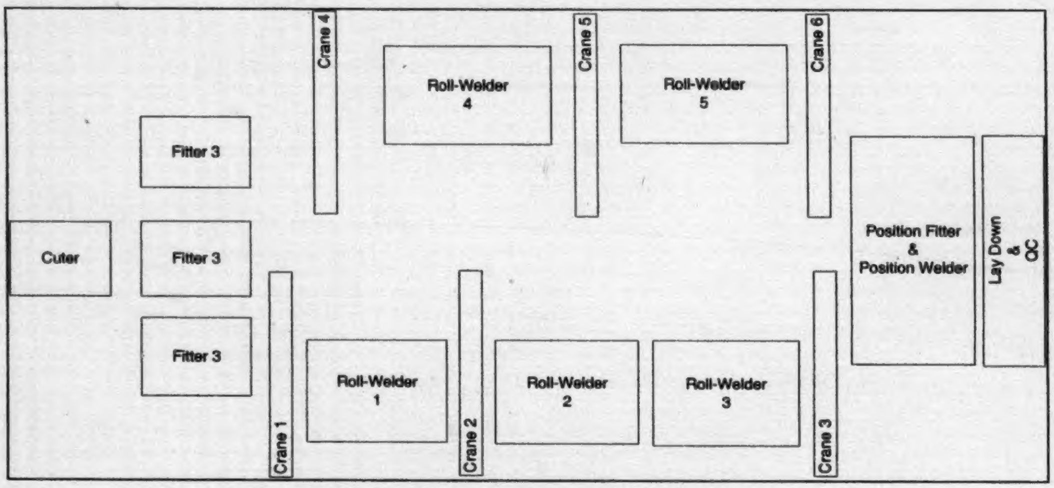

Crane 4
Crane 5
Crane 6
Roll-Welder 4
Roll-Welder 5
Fitter 3
Cuter
Fitter 3
Fitter 3
Position Fitter & Position Welder
Lay Down & QC
Roll-Welder 1
Roll-Welder 2
Roll-Welder 3
Crane 1
Crane 2
Crane 3

Figure 2-2. Layout of the Old Shop

Shop drawings were issued into the shop batch by batch. The cutter cut pipes based on shop drawings and sent them to the central floor. Roll-fitters walked to the floor to pick up the cut pipes and fittings, and then moved the cut pipes to fitting stations, sometime using cranes, if necessary. After grinding and fitting the pipes, the roll-fitter moved the fitted pipes back to the central floor. Roll-welders walked to the floor to pick up any one of the fitted pipes, moved it to the roll-welding stations, again, using cranes if necessary. Roll-welders welded the fitted pipe, and then moved it back to the central floor. In most cases, the parts need more than one incidence of roll-fitting and roll-welding. Therefore, roll-fitters might have needed to move the parts from the central floor back to the roll-fitting stations again to do roll-fitting for a second time. This was also the case for the roll-welders. The number of repetitions depended on the configuration of each spool. The position-fitter was located at the end of the shop. He kept looking for all welded parts for one spool from the central floor based on shop drawing and moved them one by one to the position-fitting station. After finding all the parts, he could start fitting them together into a spool. After fitting, a position welder welded at the same station. The finished spools were tested by the quality checking crew, and then they were moved to the floor to await shipment out of the shop for the next process.

In order to analyze the old system and identify the non-value-added activities, the production management staff drew a value stream map (VSM) for the old system (Figure 2-3). Because of the unique nature of pipe spool and the resulting wide range of uncertainty, the staff can only estimate loosely for each activity and inventory. The estimated minimum cycle time is 37 minutes (the summary of minimum time of each activity and inventory) and the maximum cycle time is 42 days (the summary of maximum time of each activity and inventory) based on the VSM.

The minimum cycle time and the maximum cycle time indicate two extreme cases. In reality, they might happen with very low possibility. This VSM failed to model material handling, to link diameter inch with the activity duration, and to model the dynamic nature of the shop.

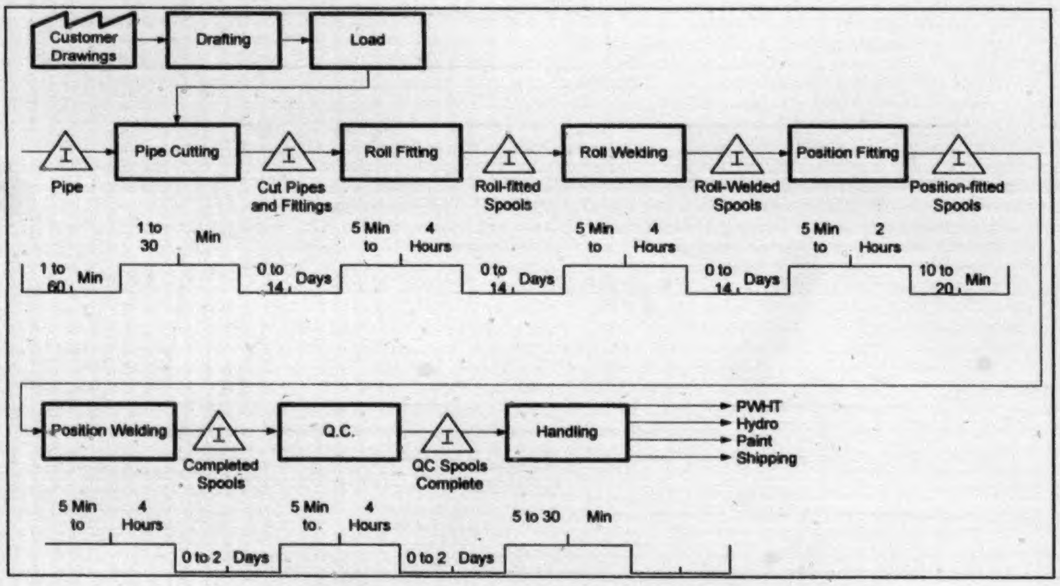

Customer Drawings
Drafting
Load
Pipe Cutting
Roll Fitting
Roll Welding
Position Fitting
Pipe
Cut Pipes and Fittings
Roll-fitted Spools
Roll-Welded Spools
Position-fitted Spools
1 to 30 Min
5 Min to 4 Hours
5 Min to 4 Hours
5 Min to 2 Hours
1 to 60 Min
0 to 14 Days
0 to 14 Days
0 to 14 Days
10 to 20 Min
Position Welding
Q.C.
Handling
PWHT
Hydro
Paint
Shipping
Completed Spools
QC Spools Complete
5 Min to 4 Hours
5 Min to 4 Hours
5 to 30 Min
0 to 2 Days
0 to 2 Days

Figure 2-3. Value Stream Map of Old System

The old fabrication system used an activity-oriented layout. The machines and laborers that had the same function were grouped and sequenced from the entrance to the exit of the shop. Several wastes were identified in this configuration. High inventory was a property of this system. The floor was always cluttered. Cut pipes, fitted parts, and welded parts were piled on the floor. The waiting time of parts is long, especially the roll-welded parts waiting for position-fitting. Therefore, a lot of space in the shop was occupied. Roll fitters and roll welders were scattered throughout the shop. Materials were moved between working stations and the central floor repeatedly. Motion is also a type of waste. "Motion" refers to the extra steps taken by employees and equipment to accommodate inefficient process layout, defects, reprocessing, overproduction, and excessive or insufficient inventory. In this system, fitters and welders had to walk back and forth with transportation of fabricated parts between the working stations and the central floor. The position-fitter even spent time looking for the required parts of a spool on the central floor and sometimes returned without finding the required parts. A scheduling problem was also associated with the old system. There was no consistent queuing rule to follow. The roll-welders randomly picked fitted parts to work on and the position-fitter randomly picked roll-welded parts to work on. This resulted in some materials and parts staying on the floor for a long time. The final result of the above waste and the queuing problem was a long cycle time of spool fabrication.

2.4.2 Flow Spool Fabrication Shop

Because of the high uncertainty and complexity of spool fabrication, the fabricator decided to test lean production in only one of the five shops in order to reduce risk. The fabricator changed its traditional fabrication shop layout to a flow fabrication system. In order to facilitate a flow system, other techniques such as 5S (Sort [Seiri], Set in Order [Seiton], Shine [Seiso], Standardize [Seiketsu], and Sustain [Shitsuke]) and virtual factory, were also applied. Figure 2-4 shows the new layout, where one cutter is at the front of the shop and five work cells are arranged along the two sides. There is one fitter and one welder in every work cell. Laborers were cross-trained so that each fitter could do both roll-fitting and position-fitting and each welder could do both roll-welding and position-welding.

Shop drawings are loaded into the shop batch by batch. Cutters cut pipes and send them to the central floor. Drawings are assigned to the five work cells according to the length of the spool. The maximum quantity that each work cell can hold is 200 welding diameter inches. The fitter of each work cell moves all the pipes and fittings composing one spool from the central floor to the work cell using cranes if necessary. A spool is completely fabricated in one work cell. The queuing rule, First In First Out (FIFO), is strictly followed if there is more than one spool being fabricated in the work cell. The fitter sends the finished spools to the lay down area, where they are checked by the QC crew, and then moved out of the shop by the mobile crane.

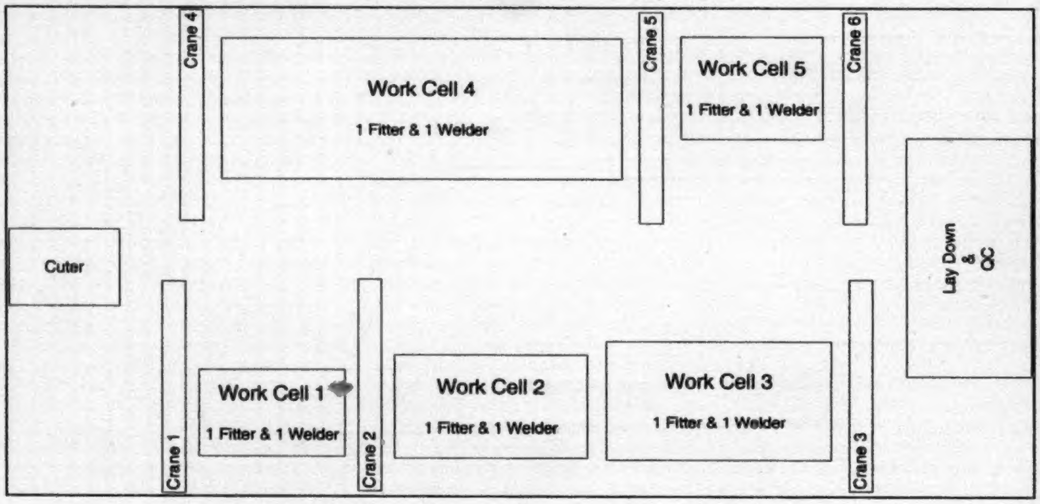

Crane 4
Work Cell 4
1 Fitter & 1 Welder
Crane 5
Work Cell 5
1 Fitter & 1 Welder
Crane 6
Cuter
Lay Down & QC
Work Cell 1
1 Fitter & 1 Welder
Work Cell 2
1 Fitter & 1 Welder
Work Cell 3
1 Fitter & 1 Welder
Crane 1
Crane 2
Crane 3

Figure 2-4. Layout of the New Shop

The value stream map for the newly reconfigured system was drawn as well (Figure 2-5). The minimum cycle time is 22 minutes (the summary of minimum time of each activity and inventory) and the maximum cycle time is 25 days (the summary of maximum time of each activity and inventory) based on the VSM.

Similar with the old shop, the minimum cycle time and the maximum cycle time of the new shop indicate two extreme cases. The VSM of the new shop also failed to model material handling, to link diameter inch with the activity duration, and to model the shop dynamic nature.

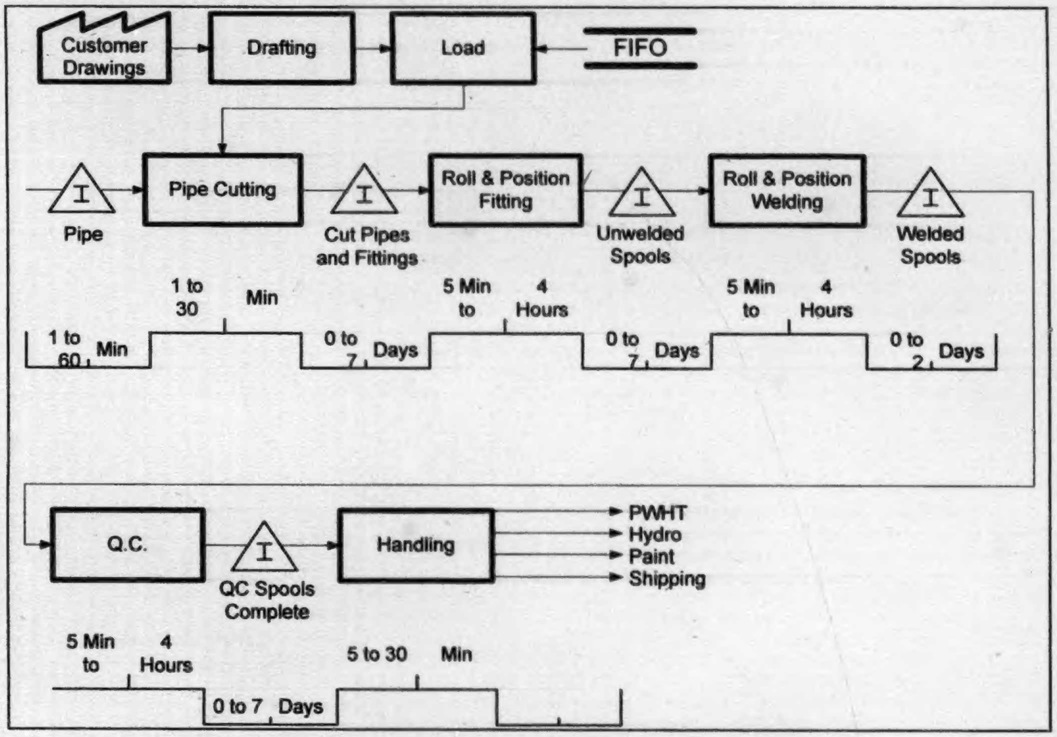

Customer Drawings
Drafting
Load
FIFO
Pipe Cutting
Pipe
Roll & Position Fitting
Cut Pipes and Fittings
Roll & Position Welding
Unwelded Spools
Welded Spools
1 to 30 Min
5 Min to 4 Hours
5 Min to 4 Hours
1 to 60 Min
0 to 7 Days
0 to 7 Days
0 to 2 Days
Q.C.
Handling
QC Spools Complete
PWHT
Hydro
Paint
Shipping
5 Min to 4 Hours
5 to 30 Min
0 to 7 Days

Figure 2-5. Value Stream Map of the New System

In the new system, a flow is formed in each work cell. All roll-fitting, roll-welding, position-fitting, and position-welding are sequenced and finished in one work cell. Evaluated by lean thinking again, the above-identified wastes are reduced or eliminated. Inventory is greatly reduced in the new system. Fitted or welded parts are no longer piled on the central floor. The wait is reduced, as is the inventory. The time that roll-welded parts must wait for position-fitting is also greatly reduced owing to the elimination of the bottleneck for position-fitting. The space used for inventory is thereby reduced. The floor becomes clearer and more orderly. In the new system, most of the material handling happens inside the work cell. The repetitive movement of materials between working stations and the central floor is eliminated. The transport distance between fitting and welding is reduced to almost zero. Associated with a reduction in transport, the extra motion of fitters and welders between the working station and the central floor is eliminated as well. The FIFO queuing rule is consistently followed if there is more than one spool being fabricated in each work cell. This guarantees that no part or spool stays in the work cell for very long. In summary, a shorter fabrication cycle time is achieved by the reduction or elimination of waste and by following the FIFO queuing rule.

2.5 WEAKNESS OF VALUE STREAM MAP

As discussed above, some improvements in shop performance have been observed. The value stream map was used as a tool to describe and compare the two systems in order to facilitate the implementation of the approach. As a paper and pen tool, the value stream map is easy to learn and may be used to identify waste in a system. However, since the value stream map is a static snapshot and is constructed by going along the flow, it cannot adequately represent the variability, dynamic nature, and high uncertainty of such a complex system with product uniqueness and high mix. The comparison of the VSM and simulation was discussed by Lian and Van (2002), and similar findings were obtained. In the system studied in this research, every spool is unique and vastly different in terms of size and configuration. The activity duration and the inventory delay of individual spools vary widely. Fitting and welding activities may be repeated a few times as determined by the configuration of the spool. The system is frequently interrupted by rework or by a client's imminent need. The cycle time of a spool may range from several minutes to a few weeks. It is very difficult for production staff to utilize a value stream map to model and represent such a system when evaluating the improvements made by a new system. This is demonstrated in historical data analysis and the simulation results published in a later section of this chapter.

We need a more powerful tool to model such a system, to evaluate the performance changes, and to test more scenarios. In this research, the simulation-based approach is chosen as the tool best able to evaluate the application of lean techniques in industrial construction fabrication.

2.6 SIMULATION-BASED APPROACH

Simulation is widely used in the manufacturing industry. In the construction industry, there have been a lot of studies on repetitive construction processes using simulation for the past twenty years. Simulation has been proven to be a powerful tool to model and analyze production processes. Lean thinking, in a similar way, focuses on studying production processes in order to eliminate waste and to improve quality. As Halpin remarks, "Lean thinking and simulation are very closely linked and even synonymous" (2002). Simulation can be used as a quantitative means to test and validate lean concepts and applications prior to implementation. For example, Tommelein (1998) successfully used simulation to model a matching-problem to compare push-driven and pull-driven processes. The simulation results showed that the pull-technique in material delivery is a useful tool for improving performance. This paper uses simulation to facilitate the application of flow production to improve the performance of pipe spool fabrication.

2.6.1 Choosing the Modeling Tool and Developing New Elements

In order to describe and compare the two systems and to experiment with alternative scenarios of the new system, the pipe spool fabrication shop is modeled using the enhanced Common Template in the simulation development environment *Simphony* (Hajjar and AbouRizk 2002). *Simphony*'s Common Template is a general-purpose simulation (GPS) tool for discrete-event simulation. Table 2-1 summarizes the functionality of several important modeling elements in the Common Template (*Simphony* User's Guide 2000).

Table 2-1. Selected Modeling Elements of Common Template in *Simphony*

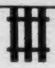
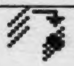
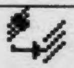
NAME	NOTATION	DESCRIPTION
Composite Element	■	It is used to build sub-models inside the main model. The user can create other elements as children inside a composite element.
Create Entities	■	It creates new entities by the number and time intervals specified by the user and transfer them out.
Set Entity Attributes	■	It assigns values for new or existing attributes of entities passing through it.
Declare Resource	■	It defines one type of resource in the model in addition to the available number of units.
Waiting File	■	It defines a queue for entities. The entities waiting in the file will be ranked according to the priority associated with each of them.
Task	■	It represents a normal task that requires a duration to perform. Each entity transferred to the element is transferred out after the delay time specified in the

		input parameters of the task element.
Capture Resource	■	It is triggered by any entity that is transferred into it. Upon the arrival of an entity, the element adds it to the file defined by user. A check for the entities in that file is then triggered at the file element. Each capture element can be assigned a capture priority number. This number specifies how the entities should be ranked in the waiting file. The higher the number, the higher the priority of the entity to get its requested resources.
Release Resource	■	It is triggered by any entity transferred into it. Upon the arrival of an entity, the element navigates all its child elements and frees resources defined in each of these children. If no child elements exist, the element automatically releases all the resources allocated to the incoming entity.
Conditional Branching	■	It enables the routing of entities into two different branches based on a condition associated with the element. The condition is based on an attribute of the current entity passing the element. If the condition is true, the entity will be routed through the true branch, and if false it will go through the false branch.
Probabilistic	■	It enables the routing of entities into a number of

Branching		different branches based on a probability associated with each branch.
Consolidate	■	It helps to manage the number of entities flowing through it. The element has two output connection points. The right-hand side output connection point transfers out the same entities that are transferred into the element while the bottom output point transfers out clones of these elements depending on the setting of the element. The element waits until the specified number of entities to consolidate is reached then it produces the specified number of entities to generate. The generated entities are clones of the last entity passing the element when the number to consolidate was reached.

In spool fabrication, every spool is unique and needs to be recognizable during the whole process. Moreover, each spool does not travel through the system as one entity most of the time. Rather, it flows in the form of raw materials or a few parts. In order to address this, three new simulation elements, “Assembly”, “Batch” and “Unbatch” were designed and developed by the author to enhance the common template of *Simphony* for modeling these circumstances. Table 2-2 summarizes the functions of the newly developed elements.

Table 2-2. New Elements Enhancing Common Template of *Simphony*

NAME	NOTATION	DESCRIPTION
Assembly		Entities with the same product ID can find each other in a queue in this element and transfer out as one entity. The number of entities to be assembled can be dynamic and defined by the attribute of entities.
Batch		Specified number of entities can become a bundle in this element and can transfer out as one entity, but each of them is still distinguishable, and all features of each entity are kept as in the original.
Unbatch		Always paired with the "Batch" element. Batched entities are released here to their original state without any change to each of them.

The models for both the old and new systems describe 100 spools being fabricated in the fabrication shop. In order to model the process close to reality, the configuration of the spools fabricated in this specific shop and their shop-specific production processes were studied. The study compared simulation results with real production performance.

"Diameter inch" was identified as the most critical configuration data of a spool. Diameter inch of a spool, in this context, is the total welding diameter inch for the spool. For the purposes of comparing systems, it is assumed that diameter inch is the dominant factor to determine the production time of each main activity of cutting, fitting, and welding. The data for spools fabricated in this shop in the last two years were extracted from a database, and based on this data, the diameter inch distribution was fitted. The distribution was used to generate the diameter inch of the 100 sample spools. A spool has two other important parameters that capture the features of its production process: the "Roll-welding Parts Number" is the number of the decomposed parts, each of which can be roll-fitted and roll-welded; and the "Rolling-welding Repeating Times of Each Part" is the repeating time of roll-welding of each part. This information was deduced from the production crew's experience and generated as a random distribution in the model. The time study of cutting, fitting, welding, and materials handling was conducted by time recording in shop and using interviews with superintendents and foremen.

2.6.2 Simulation Model for Batch-and-Queue Fabrication Shop

Figure 2-6 shows the old fabrication system, with the model depicted at the top of the hierarchy. This level of the hierarchy models both the shop layout and the process flow. Each working station shown at this level is detailed by a process model simulating the activities and resource interactions within this working station. Detailed process models for a roll fitting station and roll welding station are demonstrated in Figure 2-7 on the second level of the hierarchy. This two-level hierarchical graphic model virtually maps the whole production system described in Section 2.4.1.

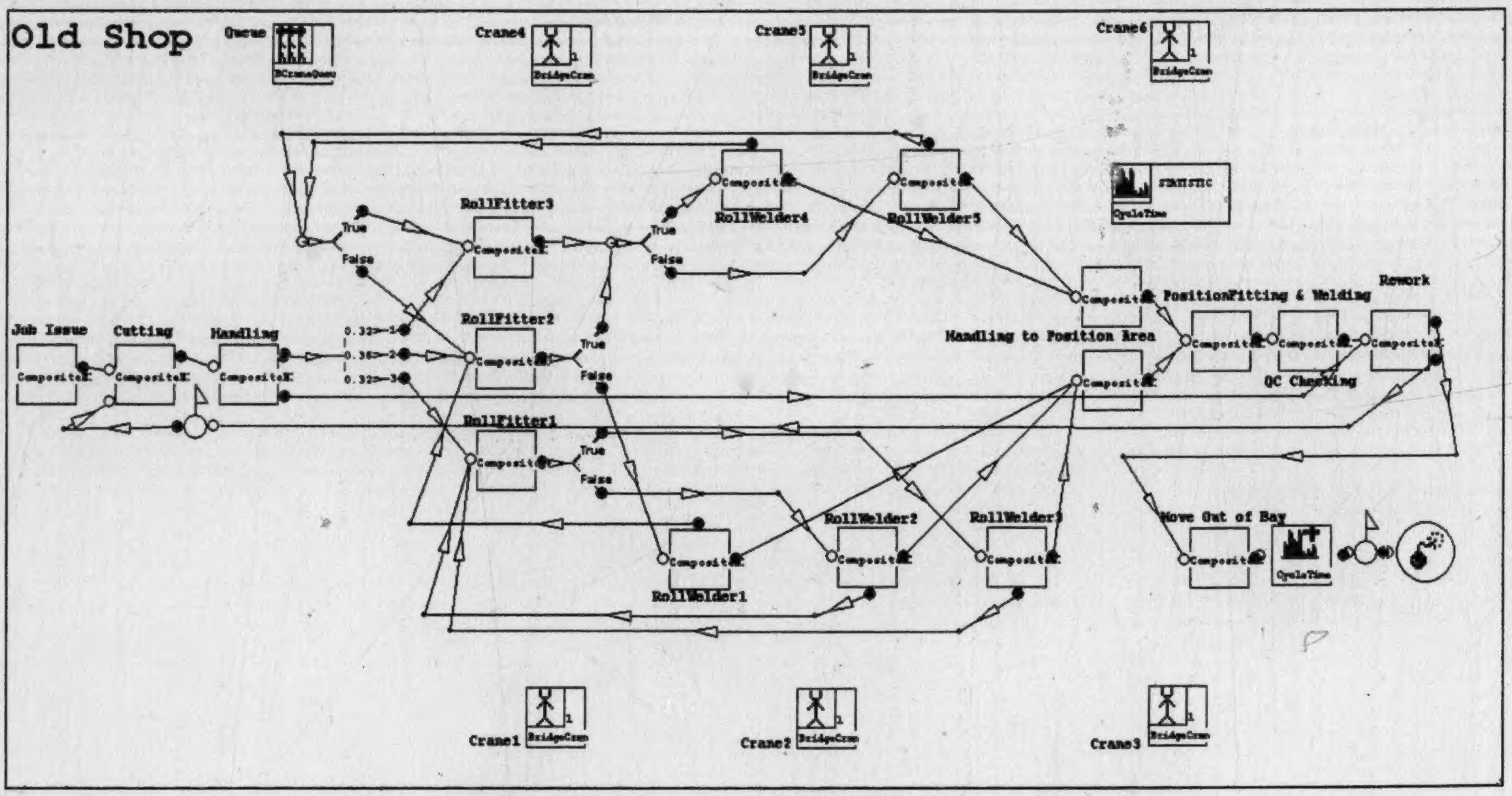

Old Shop
Queue
DCraneQueu
Crane4
BridgeCran
Crane5
BridgeCran
Crane6
BridgeCran
STATISTIC
CycleTime
True
False
RollFitter3
CompositeX
True
False
RollWelder4
RollWelder5
Job Issue
Cutting
Handling
CompositeX
0.32>-1
0.36>-2
0.32>-3
RollFitter2
True
False
RollFitter1
True
False
Handling to Position Area
PositionFitting & Welding
Rework
QC Checking
RollWelder1
RollWelder2
RollWelder3
Move Out of Bay
CycleTime
Crane1
BridgeCran
Crane2
BridgeCran
Crane3
BridgeCran

Figure 2-6. Simulation Model of Batch-and-Queue Fabrication Shop

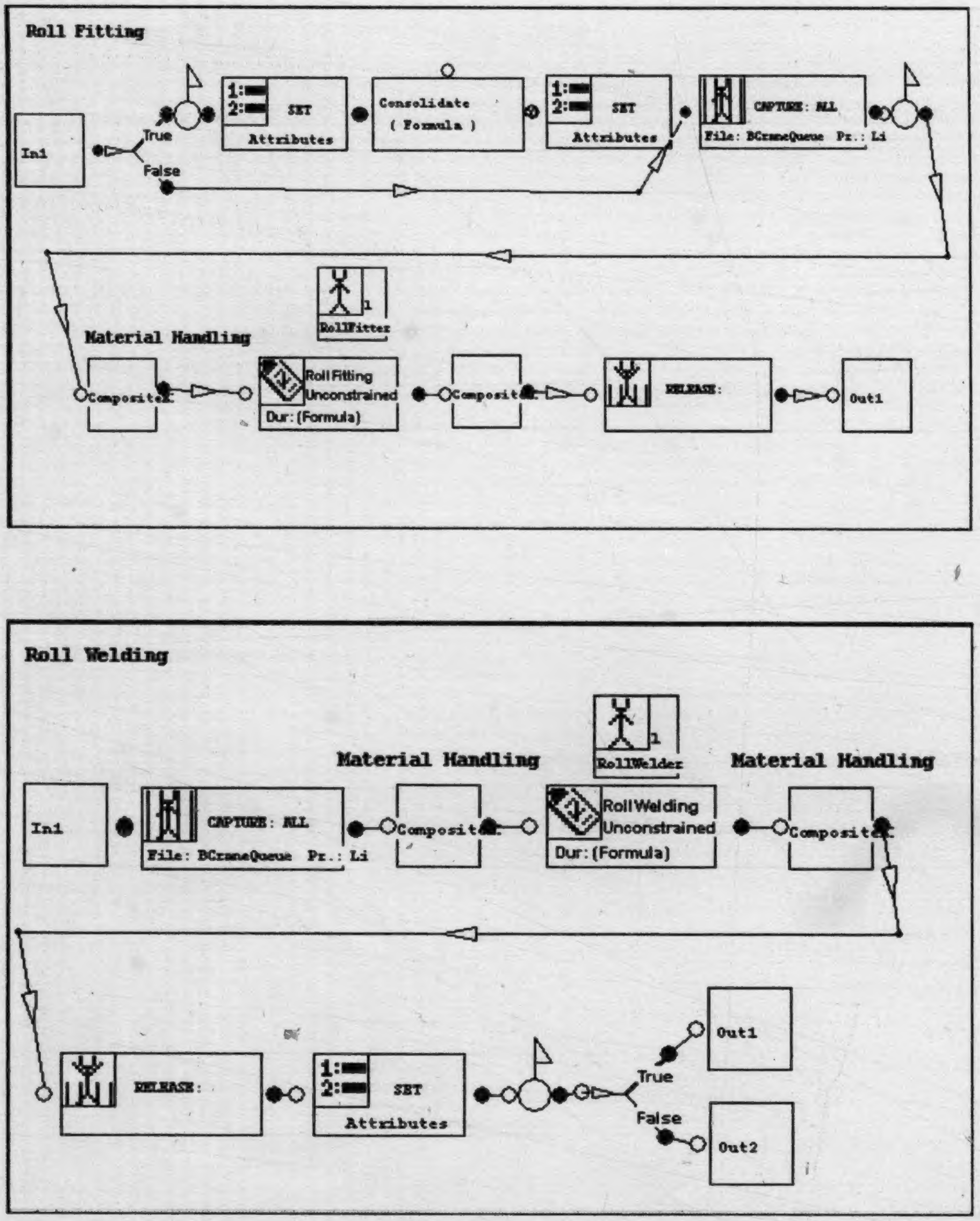

Roll Fitting
In1
True
False
1:
2:
SET
Attributes
Consolidate
(Formula)
1:
2:
SET
Attributes
CAPTURE: ALL
File: BCraneQueue Pr.: Li
RollFitter
1
Material Handling
Composite
Roll Fitting
Unconstrained
Dur: (Formula)
Composite
RELEASE:
Out1
Roll Welding
Material Handling
RollWelder
1
Material Handling
In1
CAPTURE: ALL
File: BCraneQueue Pr.: Li
Composite
Roll Welding
Unconstrained
Dur: (Formula)
Composite
RELEASE:
1:
2:
SET
Attributes
True
False
Out1
Out2

Figure 2-7. Detailed Process Model of a Roll Fitting Working Station and a Roll Welding Working Station

Data collection was conducted in order to populate the developed model. Collection tasks include:

- Extracting spool fabrication information from the contractor's information system, analyzing 2500 spools fabricated in the studied shop; fitting statistical distribution of diameter inch of these spools; and calculating the percentage of module spool and non-module spool out of total spools. These describe the typical spools fabricated in this shop.
- Gaining fabrication-related spool configuration information by interviewing managers, superintendents, and foremen. This information includes parts amounts, the repeat times of rolling welding for part, the total times of position welding. This type of information relied on the experience of industry management staff.
- A time study of all activities in the fabrication shop, such as cutting, fitting, welding, and material handling. The information was collected from the time study of the contractor, the estimations of shop management staff, and the time study by the author.

Collected data are summarized in Table 2-3.

Table 2-3. Data Collection Summary for Modeling Old System

PARAMETER		VALUE
Total Diameter Inch (DI)		Exponential (10.68)
Module Spool (30%)	Total Number of Parts for RollWelding / PartsAmount (PA)	Triangular (1,3,6)
	Repeat Times of RollWelding For Each Part (RTsRWP)	Triangular (1,2,3)

	Total Times of PositionWelding (TsPW)-Final Assembly	1
Non-Module Spool (70%)	Total Number of Parts for RollWelding / PartsAmount (PA)	Triangular (1,2,3)
	Repeat Times of RollWelding For Each Part (RTsRWP)	Uniform (1,2)
	Total Times of PositionWelding (TsPW)-Final Assembly	1
Total Cutting Time for One Spool	Module Spool	1.2*DI
	Non-Module Spool	1.2*DI
Material Handling Time	From cutting station to floor without crane (60%) (Average total time for one spool)	Triangular (1, 2.5, 5)
	From cutting station to floor with crane (40%) (Average total time for one spool)	Triangular (3, 7, 10)
	Anywhere inside the bay	Triangular (5, 10, 15)
RollFitting	Total RollFitting Time (TRFT)	(7.45*DI)*0.9
	RollFitting Time of Every Time	TRFT/(PA*RTsRWP)
RollWelding	Total RollWelding Time (TRWT)	(4.584*DI)*0.9
	RollWelding Time of Every Time	TRWT/(PA*RTsRWP)
PositionFitting	Total PositionFitting Time (TPFT)	(10.45*DI)*0.1
PositionWelding	Total PositionWelding Time (TPWT)	(11.91*DI)*0.1

QC Checking Time (Visual, PMI, LPI, UT)		Triangular (120, 360, 480)
Time of Moving Spools Out of Bay	Single (50%)	Triangular (5, 13, 30)
	Bundle (50%)	

Note:

1. If the parameter is time, the unit of measurement is in minutes.

2. Cutting, fitting, and welding times were obtained base on carbon steel pipes with 12" diameter and 0.5" thickness.

3. The statistical distribution of diameter inch was fitted using BestFit based on 2500 spools fabricated in this shop.

A sample job consisting of 100 spools was loaded into the bay. The model is run and the time required to produce each of the 100 spools was recorded in the simulation. The recorded data were extracted from the simulation database and were analyzed using BestFit. The average simulation fabrication cycle time of a spool is 10.8 working days. The maximum cycle time is 55.42 days. The cycle time distribution is given in Figure 2-8.

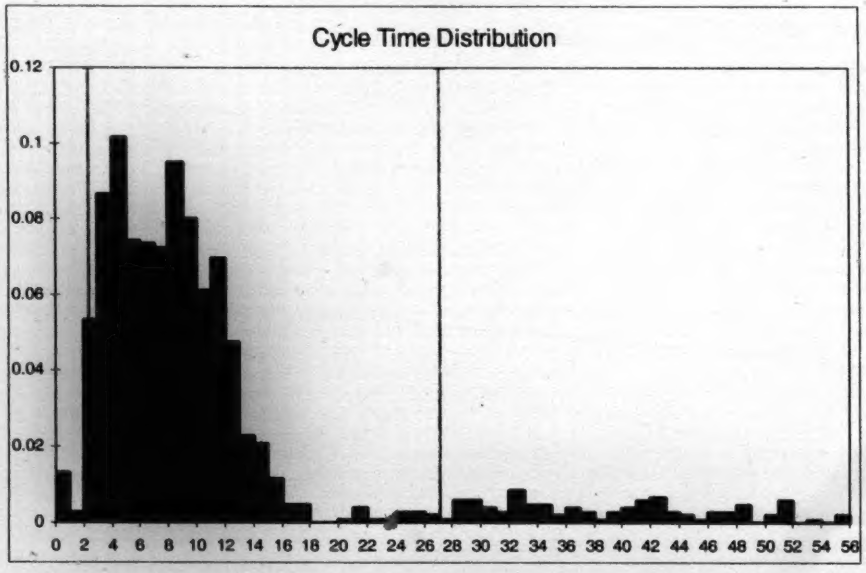


Figure 2-8. Cycle Time Distribution of Simulation Output of Batch-and-Queue Fabrication Shop

2.6.3 Simulation Model for Flow Fabrication Shop

Figure 2-9 shows the new fabrication system with the model depicted at the top of the hierarchy. This level of the hierarchy also models both the shop layout and the process flow. Each working station shown at this level is detailed by a process model simulating the activities and resource interactions within this working station. A detailed process model for a work cell is demonstrated in Figure 2-10 on the second level of the hierarchy. This two-level hierarchical graphic model virtually maps the whole production system described in Section 2.4.2.

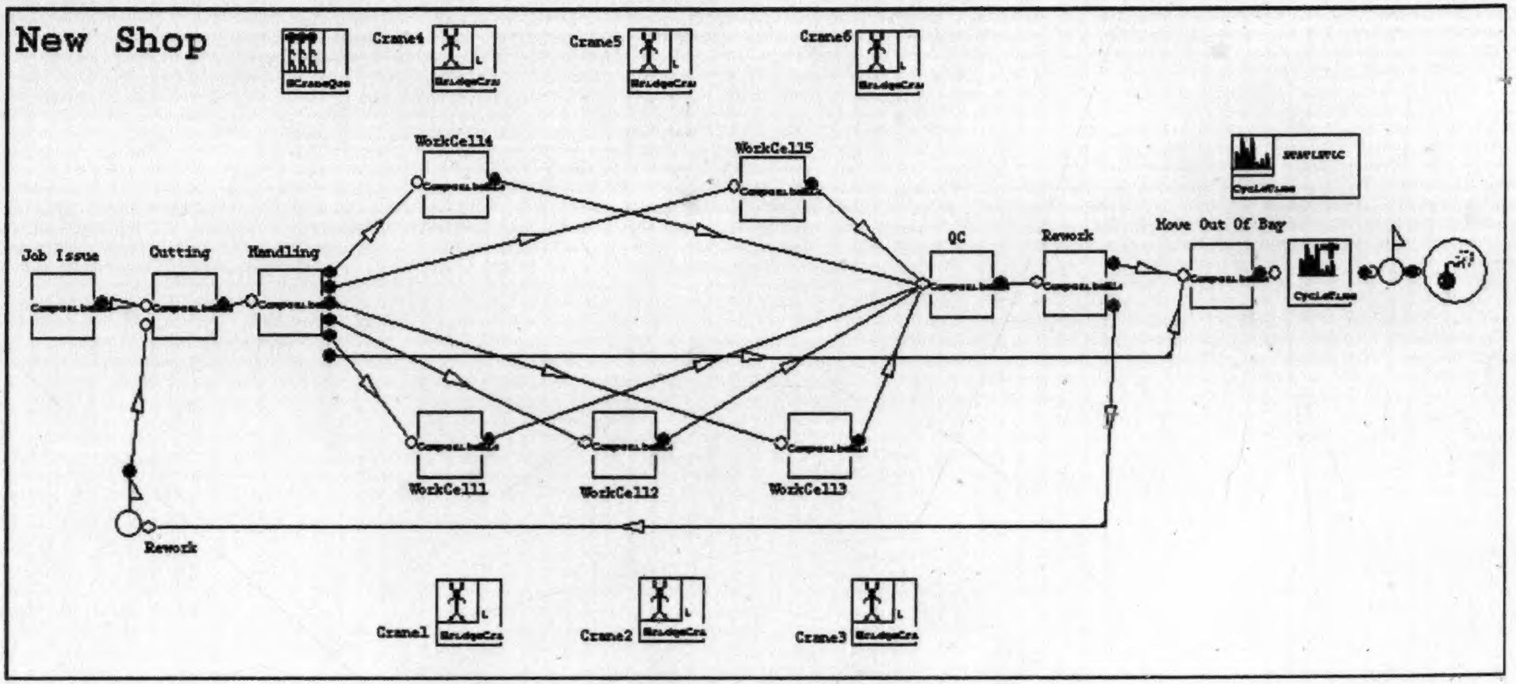
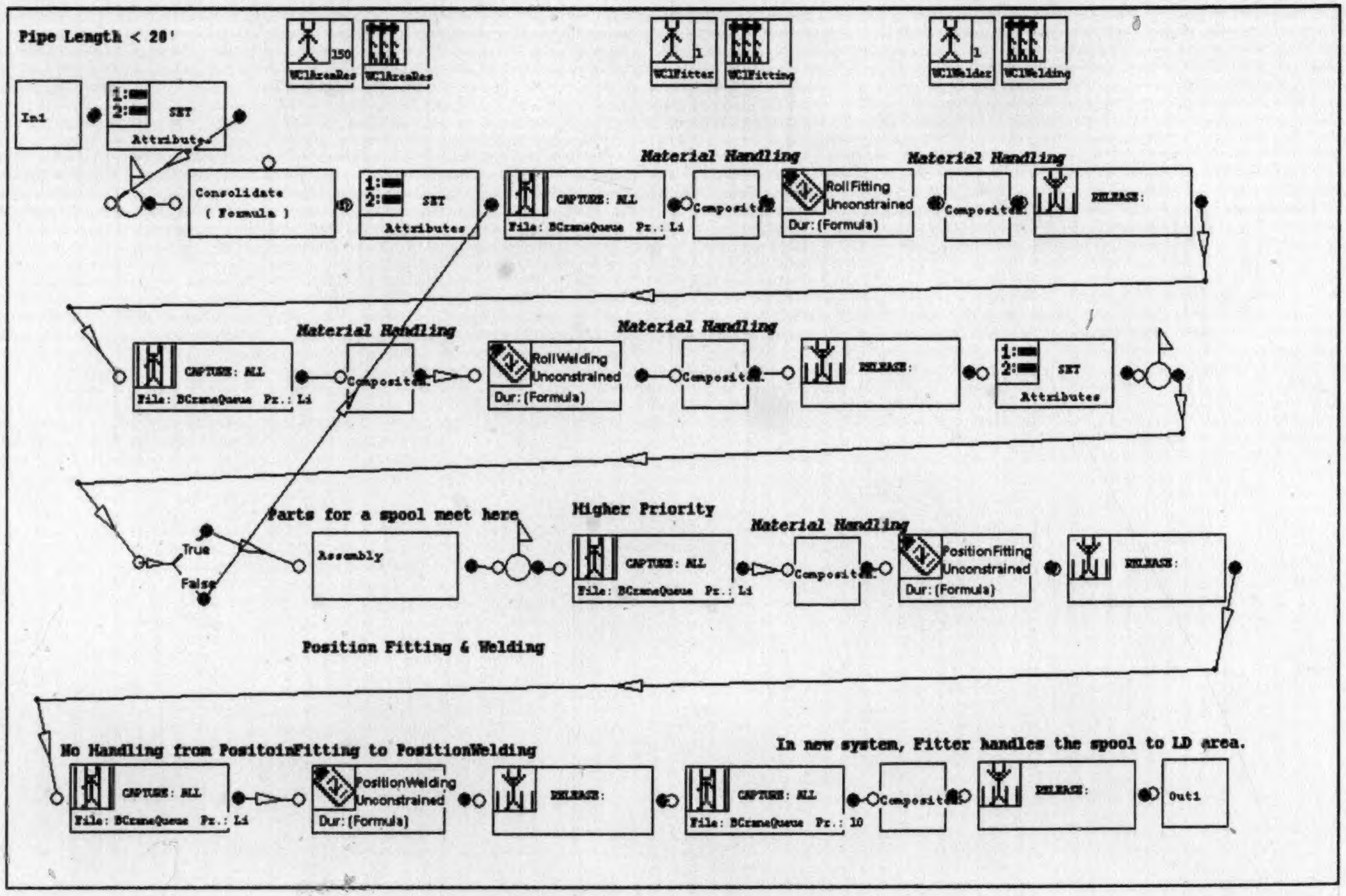

New Shop
Crane4
Crane5
Crane6
WorkCell4
WorkCell5
Job Issue
Cutting
Handling
QC
Rework
Move Out Of Bay
CycleTime
WorkCell1
WorkCell2
WorkCell3
Rework
Crane1
Crane2
Crane3

Figure 2-9. Simulation Model of Flow Fabrication



Pipe Length < 20'
150
WC1AreaRes
WC1AreaRes
1
WC1Fitter
WC1Fitting
1
WC1Welder
WC1Welding
Ini
1:
2:
SET
Attributes
Consolidate
(Formula)
1:
2:
SET
Attributes
Material Handling
CAPTURE: ALL
File: BCraneQueue Pr.: Li
Composite
Roll Fitting
Unconstrained
Dur: (Formula)
Material Handling
Composite
RELEASE:
Material Handling
CAPTURE: ALL
File: BCraneQueue Pr.: Li
Composite
Roll Welding
Unconstrained
Dur: (Formula)
Material Handling
Composite
RELEASE:
1:
2:
SET
Attributes
Parts for a spool meet here
True
False
Assembly
Higher Priority
CAPTURE: ALL
File: BCraneQueue Pr.: Li
Material Handling
Composite
PositionFitting
Unconstrained
Dur: (Formula)
RELEASE:
Position Fitting & Welding
No Handling from PositoinFitting to PositionWelding
CAPTURE: ALL
File: BCraneQueue Pr.: Li
PositionWelding
Unconstrained
Dur: (Formula)
RELEASE:
In new system, Fitter handles the spool to LD area.
CAPTURE: ALL
File: BCraneQueue Pr.: 10
Composite
RELEASE:
Out1

Figure 2-10. Detailed Process Modeling of a Work Cell

Data collection was conducted in order to populate the developed model. Collection tasks are same as those described in Section 2.6.2. Collected data are summarized in Table 2-4. The information unrelated to system design has the same values as the old system.

Table 2-4. Data Collection Summary for Modeling New System

Parameter		Value
Total Diameter Inch (DI)		Exponential (10.68)
Module Spool (30%)	Total Number of Parts for Rollwelding/PartsAmount (PA)	Triangular (1,3,6)
	Repeat Times of RollWelding For Each Part (RTsRWP)	Triangular (1,2,3)
	Total Times of PositionWelding (TsPW) - Final Assembly	1
Non-Module Spool (70%)	Total Number of Parts for Rollwelding/PartsAmount (PA)	Triangular (1,2,3)
	Repeat Times of RollWelding For Each Part (RTsRWP)	Uniform (1,2)
	Total Times of PositionWelding (TsPW) - Final Assembly	1
Total Cutting Time for One Spool	Module Spool	1.2*DI
	Non-Module Spool	1.2*DI
Time of handling pipes from cutting station to floor (Total time for one spool)	Without Crane (60%)	Triangular (1, 2.5, 5)
	With Crane (40%)	Triangular (3, 7, 10)
RollFitting	Total RollFitting Time (TRFT)	7.45*(DI*90%)
	RollingFitting Time of Every Time	TRFT/(PA*RTsRWP)

RollWelding	Total RollWelding Time (TRWT)		4.584*(DI*90%)
	RollWelding Time of Every Time		TRWT/(PA*RTsRWP)
PositionFitting	Total PositionFitting Time (TPFT)		10.45*(DI*10%)
PositionWelding	Total PositionWelding Time (TPWT)		11.91*(DI*10%)
Handling Time (Handle pipes from floor to work cell)	WC 3	With 1 Crane (70%)	Normal (8, 0.8)
		With 2 Cranes (30%)	Normal (12, 0.8)
	WC 5	Without Crane (50%)	Normal (2.5, 0.2)
		With 1 Crane (50%)	Normal (6, 0.6)
Handling Time (Inside work cell)	WC 1	Without Crane (50%)	Normal (1.5, 0.2)
		With 1 Crane (50%)	Normal (5, 0.6)
	WC 2	Without Crane (40%)	Normal (1.5, 0.2)
		With 1 Crane (60%)	Normal (5, 0.6)
	WC 3	With 1 Crane (70%)	Normal (7, 0.8)
		With 2 Cranes (30%)	Normal (10, 0.8)
	WC 4	With 1 Crane (1/3)	Normal (5, 0.2)
		With 2 Cranes (1/3)	Normal (10, 0.8)
		With 3 Cranes (1/3)	Normal (13, 1)
	WC 5	Without Crane (50%)	Normal (1.5, 0.2)
		With 1 Crane (50%)	Normal (5, 0.6)
QC Checking Time	Visal		Triangular (120, 360, 480)
	PMI		
	LPI		
	UT		

	Hardness		
	X-Ray		
Time of Moving Spools Out of Bay	Single (50%)	1	Tri (5, 13, 30)
	Bundle (50%)	Uniform (2, 3, 4)	

Note:

1. If the parameter is time, the unit of measurement is in minutes.

2. Cutting, fitting and welding time were obtained base on carbon steel pipes with 12" diameter and 0.5" thickness.

3. The statistical distribution of diameter inch was fitted using BestFit based on 2500 spools fabricated in this shop.

The same sample job, consisting of 100 spools, is loaded into the shop. The model is run and the time required to produce each of the 100 spools was recorded in the simulation. The recorded data were extracted from the simulation database and were analyzed using BestFit. The average simulation fabrication cycle time of a spool is 6.7 working days. The maximum is 38.87 working days. The cycle time distribution is given in Figure 2-11.

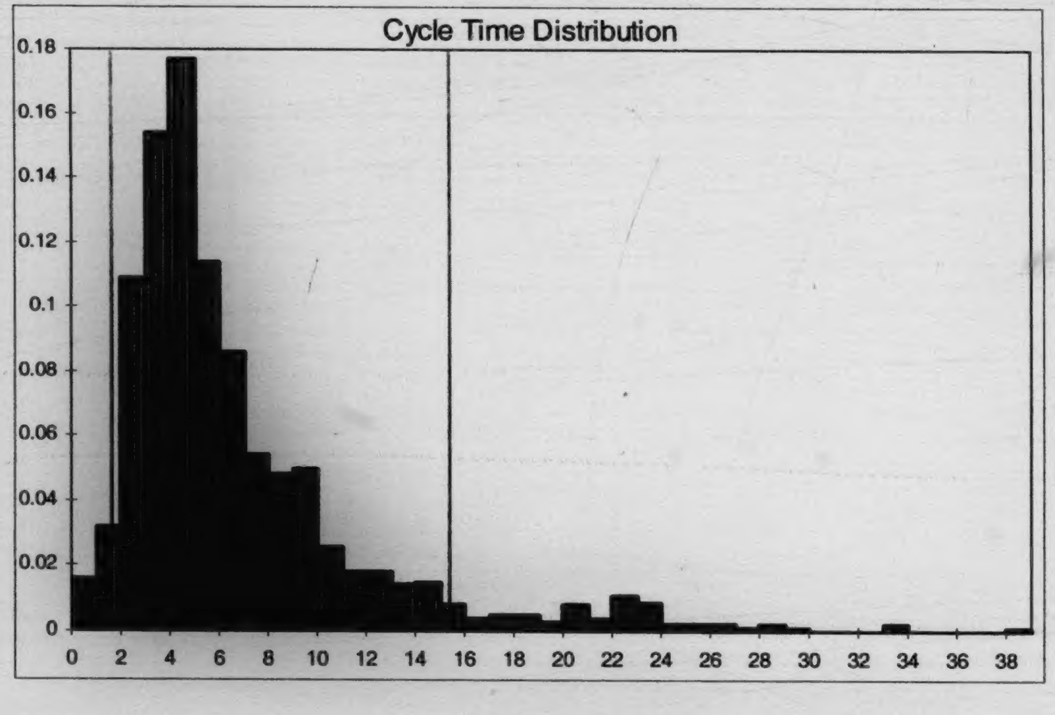


Figure 2-11. Cycle Time Distribution of Simulation Output of Flow Fabrication System

2.6.4 Historical Cycle Time Analysis

Historical data on 1146 spools fabricated in the old shop and 1161 spools fabricated in the new shop were extracted from the fabricator's information system. A template for "Change Date to Working Days" was designed in a worksheet in order to convert the raw recorded information into a useful format. The fabrication cycle time (working days) of each spool was calculated based on the designed template. Statistical analysis of the fabrication cycle time was then conducted based on the 1146 spools and 1161 spools, respectively, for the old and new systems.

The average spool fabrication cycle time of the old system is 11.5 working days. The maximum is 70 working days. The histogram of the actual working cycle time is shown in Figure 2-12.

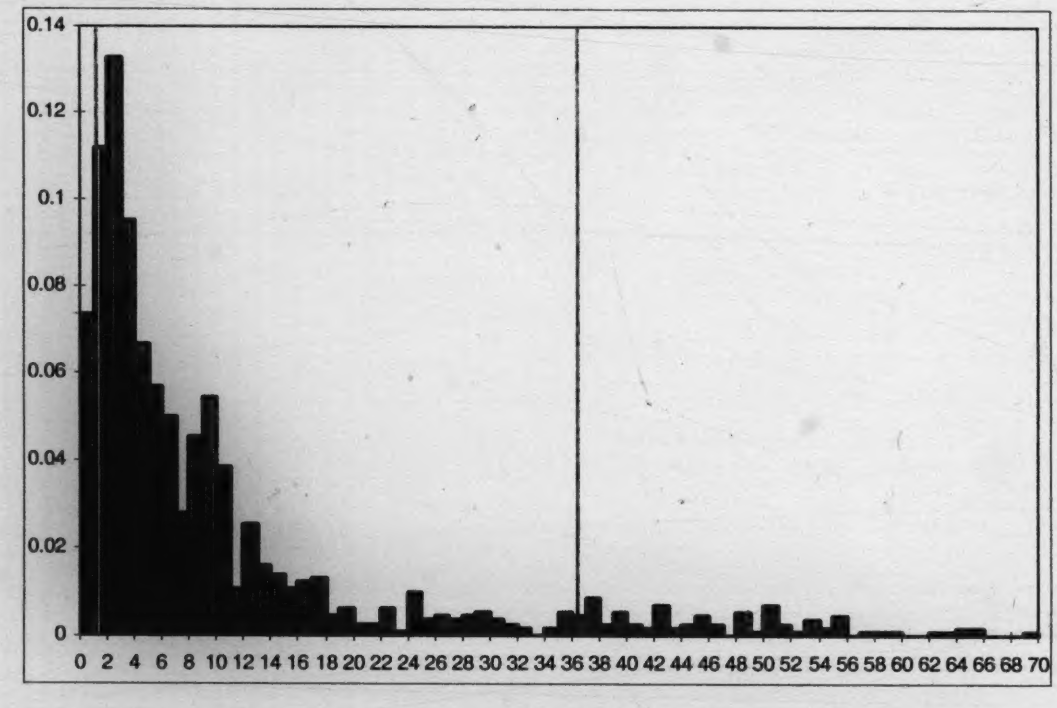


Figure 2-12. Cycle Time Distribution of Batch-and-Queue Fabrication System from Historical Data

The average spool fabrication cycle time of the new system is 7.0 working days. The maximum is 81 working days. The histogram of the actual working cycle time is shown in Figure 2-13.

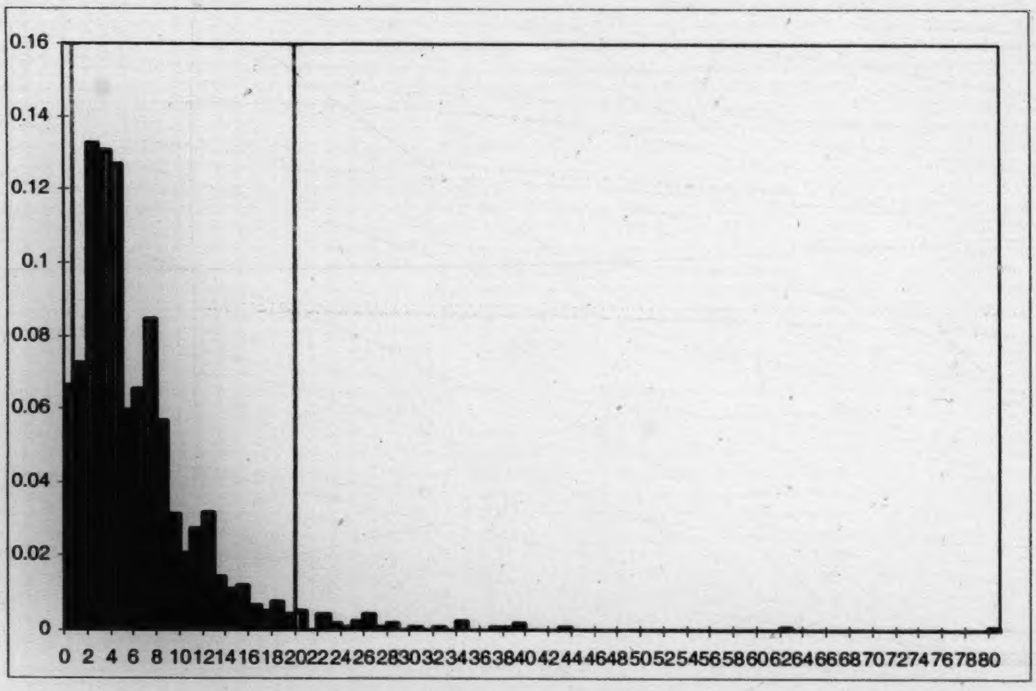


Figure 2-13. Cycle Time Distribution of Flow Fabrication System from Historical Data

2.6.5 Validation and Discussion

Comparison of Historical Cycle Time between Old System and New System

The historical cycle time analysis revealed the different features of the two systems. There is a big difference in the average fabrication cycle time of up to 4.5 days between the two systems. The cycle time distribution of the old system is more scattered. There is a certain percentage of spools with an abnormally long cycle time. The cycle time distribution of the new system, on the other hand, converges to a greater extent. The mode of the two distributions contributes to the difference as well. The likelihood of a cycle time of 2 days in the old system is higher than in the new system. The likelihood, however, of a cycle time of 3 days, 4 days, and 5 days in the new system is higher than in the old system. It is concluded that the old system has higher variability and would require more inventory on the shop floor. The new system is more controllable and carries fewer inventories on the shop floor.

Validation of New System Simulation Model and Discussion

The average simulation cycle time, 6.7 working days, is a little shorter than the average historical cycle time of 7.0 working days. The simulation cycle time distribution (Figure 2-11) and the historical cycle time distribution (Figure 2-13) are very similar with slight difference in the mode. The difference is considered acceptable. It is reasonable to assume that the average and mode values of the simulation results are smaller than that of the real result. There are a few possible explanations for this discrepancy. The model did not include certain trivial activities such as breakdowns. Although rework was simulated, the type of rework and how much time was spent on each rework was not traceable. The cycle time in the simulation model was calculated by deducting the cutting start time from the moving out time; in historical data, however, the time of issuing shop drawings, which is considered the starting time, was normally earlier than the cutting start time.

Validation of Old System Simulation Model and Discussion

The average simulation cycle time, 10.8 working days, is also shorter than the average historical cycle time of 11.5 working days. This difference is considered acceptable; however, the similarity between the simulation cycle time distribution (Figure 2-8) and the historical cycle time distribution (Figure 2-12) of the old system is not as close as in the new system. This is because the old system cannot be described as clearly. It basically followed the FIFO queuing rule for a long term run; however, the system had no clear shop control rules. During production, the queuing rule and control logic are inconsistent and prone to change. A laborer could drop all current work and change tasks to fabricate a different spool on a rush order. They could also hold some spools or forget them on the floor for a very long time. The model using the FIFO queuing rule cannot, as such, reflect the features of the old system as accurately as it does for the new system.

2.7 EXPERIMENTATION WITH THE SYSTEM USING SIMULATION

There exist more potential improvements to the studied shop and this gained experience can be introduced to the other shops as well. The simulation-based approach is used as a quantitative tool to test other scenarios.

2.7.1 Analysis of Rework Reduction

Rework can be defined as those fabrication activities that need to be redone. Rework, for the intents of this research, happens inside the shop. Shop rework should be distinguished from drawing revisions that are done before shop drawings are issued to the shop. Drawing revisions are usually caused by engineering changes. Shop rework, on the other hands, results mainly from quality problems, though scope changes may result in rework as well. Industry experience indicates that rework take place in approximately 5~10% of the total spools fabricated in a shop. Rework causes interruption of the original job sequence and scheduling, slowing down the production line. If rework is reduced, the average spool fabrication cycle time can be reduced.

The sensitivity analysis of rework reduction is conducted based on the simulation model for the new shop. A rework Composite Element is used to model the rework as shown in Figure 2-14. Rework randomly happens to the spools. The rework percentage can be changed at the second level of the hierarchy.

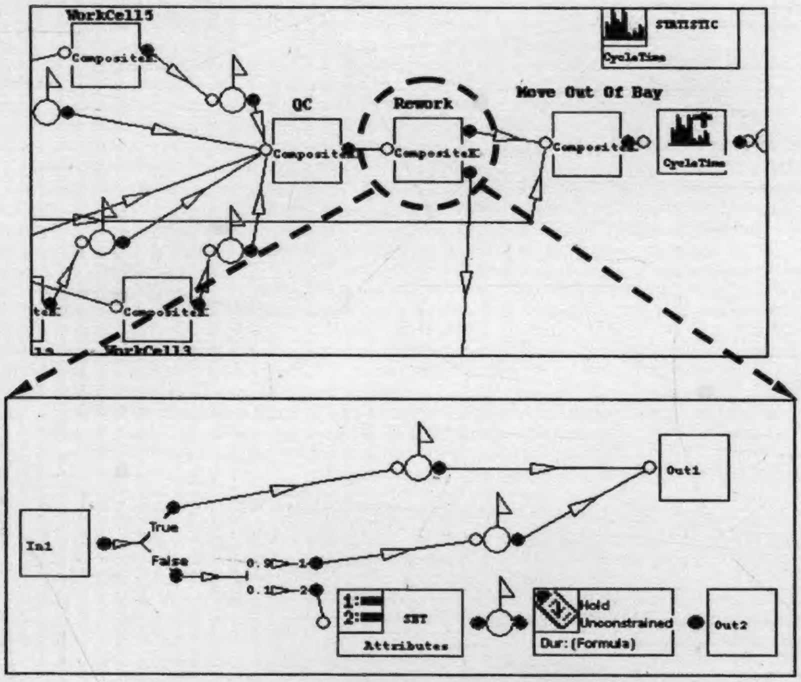

WorkCell15
CompositeK
STATISTIC
CycleTime
QC
CompositeK
Rework
CompositeK
Move Out Of Bay
CompositeK
CycleTime
CompositeK
WorkCell13
In1
True
False
0.9
1
0.1
2
Out1
1:
2:
SET
Attributes
Hold
Unconstrained
Dur: (Formula)
Out2

Figure 2-14. Modeling Rework Reduction

Five experiments are executed to test the contribution of rework reduction to average spool fabrication cycle time with the respective rework percentage of 10%, 8%, 6%, 4%, and 2%. The result is shown in Figure 2-15. It indicates that the fabrication cycle time is very sensitive to rework. Strategies should therefore be implemented to reduce rework.

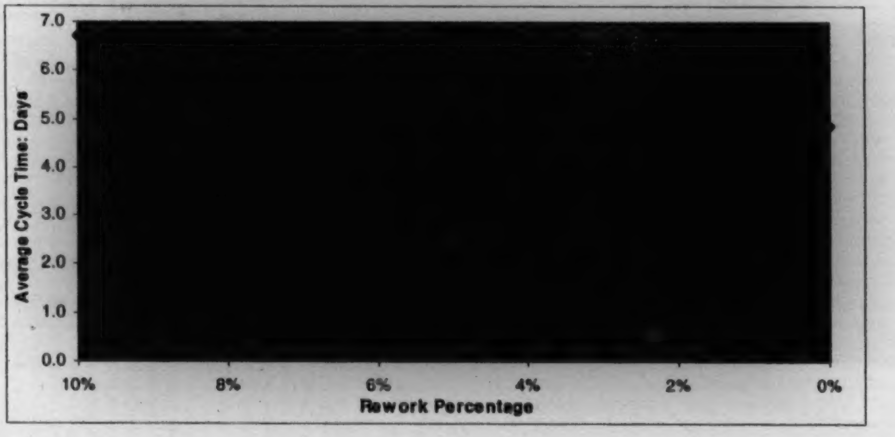


Figure 2-15. Sensitivity Analysis of Average Cycle Time to Rework Reduction

2.7.2 Analysis of Fitting Time Reduction

Cleaning and grinding are part of the fitting process, but are recognized as non-value-added activities from the perspective of lean theory. These activities are required due to an imperfect cutting process. Improving the cutting process may reduce or remove cleaning and grinding activities, which may result in a reduction of the fitting time. A search for sophisticated pipe cutting tools is conducted and a few advanced products are available in the market that can improve the cutting process quality. Nevertheless, the question remains as to the relationship between a reduction of the fitting time and the overall system performance.

The sensitivity analysis of the total fitting time reduction is conducted using the simulation model of the new shop. Both roll fitting time and position fitting time are changed in each work cell at the second level of the hierarchical model. Seven experiments are executed to test the contribution of fitting time reduction to the average spool fabrication cycle time with the respective reduction percentages of 0%, 5%, 10%, 15%, 20%, 25%, and 30%.

The results are shown in Figure 2-16. The line indicates that shorter fitting time does not always cause a shorter cycle time. The reduction of fitting time cannot proportionately reduce the average fabrication cycle time. Other critical activities, such as welding, might be the controlling factors to the shop performance. This experiment also implies the complexity of this dynamic production system. However, the overall trend of the line indicates that a reduction of fitting time is still beneficial to the average cycle time. Whether sophisticated cutting tools should be used in fabrication shops needs more cost analysis.

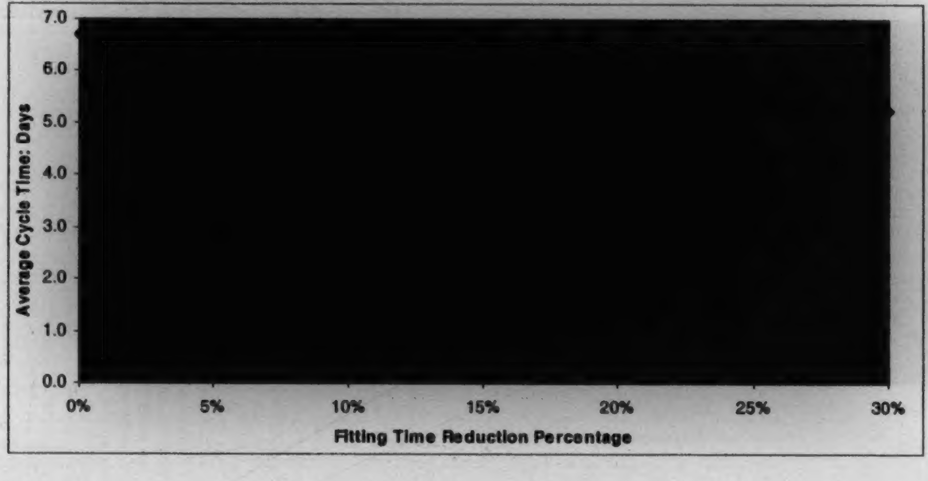


Figure 2-16. Sensitivity Analysis of Average Cycle Time to Fitting Time Reduction

2.7.3 One-Piece-Flow Fabrication

The described flow fabrication system is not yet a one-piece-flow production. The main problem is that a spool is normally decomposed into a few parts, which can be roll welded, and each part often needs to repeat the fitting and welding processes a few times.

Additionally, fitting productivity is different than welding productivity. The fitter should not be idle for a long time when the spool is being processed by the welder; each work cell processes several spools at one time. The system is therefore not an ideal one-piece-flow production.

As described in Section 4.2, fitters and welders had been cross-trained in the studied shop. If each laborer can be trained to do both fitting and welding, the work cell can be designed as a one-piece-flow production system. It also removes the non-value-added material handling between the fitter and the welder. The model of the new shop is modified based on the assumption that each laborer can do both fitting and welding, and that there are two laborers in each work cell. All process models of the five work cells are rebuilt at the second level of the hierarchical model. The simulation of one of work cells is shown in Figure 2-17.

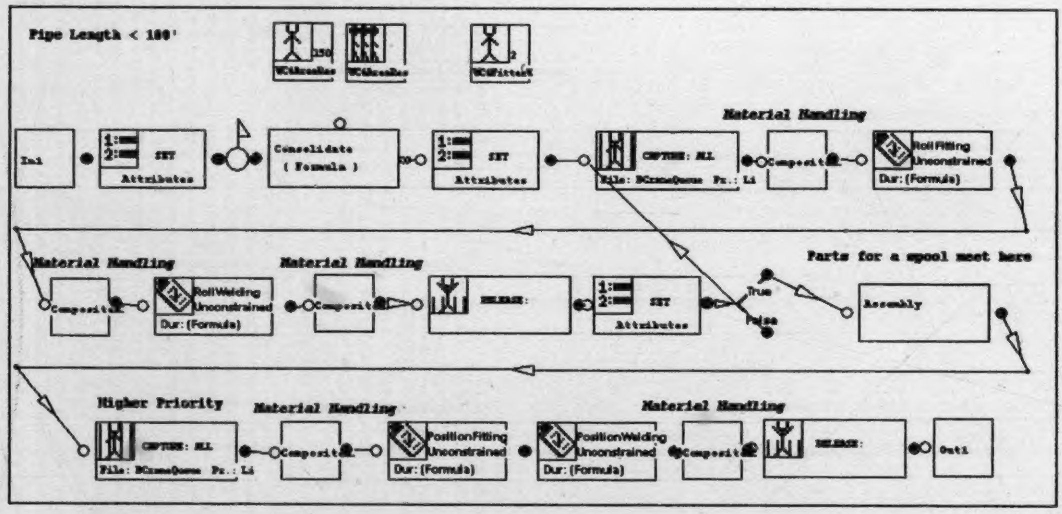

Pipe Length < 100'
150
WC4AreaRes
WC4AreaRes
2
WC4FitterX
Ini
1:
2:
SET
Attributes
Consolidate
(Formula)
SET
Attributes
Material Handling
CAPTURE: ALL
File: BCraneQueue Pr.: Li
Composite
RollFitting
Unconstrained
Dur: (Formula)
Material Handling
Composite
RollWelding
Unconstrained
Dur: (Formula)
Material Handling
Composite
RELEASE:
SET
Attributes
True
False
Parts for a spool meet here
Assembly
Higher Priority
CAPTURE: ALL
File: BCraneQueue Pr.: Li
Material Handling
Composite
PositionFitting
Unconstrained
Dur: (Formula)
PositionWelding
Unconstrained
Dur: (Formula)
Material Handling
Composite
RELEASE:
Out1

Figure 2-17. Rebuilt Model of One-Piece-Flow Production of a Work Cell

The new model is run and the simulation results analyzed. The average simulation fabrication cycle time is 4.9 working days. The maximum is 26.9 working days. The cycle time distribution is given in Figure 2-18. This is a significant improvement.

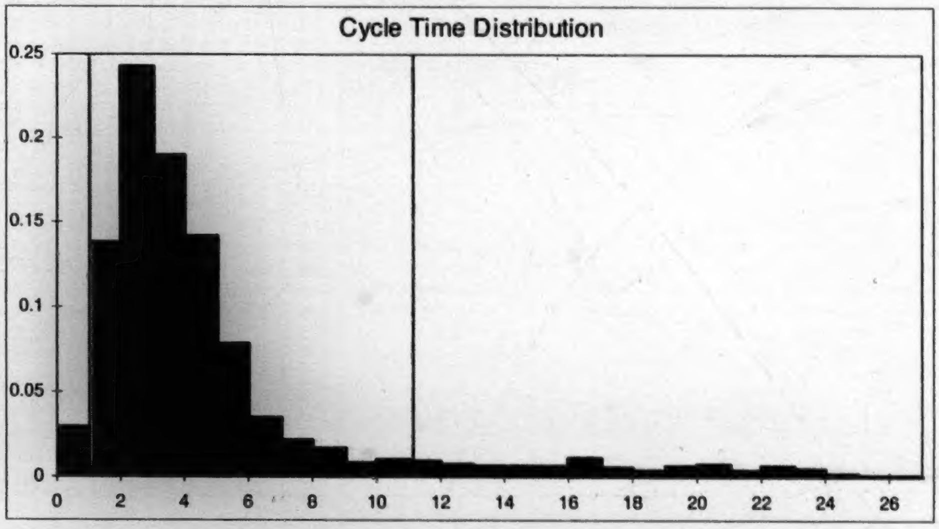


Figure 2-18. Cycle Time Distribution of Simulation Output of One-Piece-Flow Fabrication System

2.7.4 Transfer Successful Experience to Other Shops

The five shops operated by the fabricator in this research are different from each other in terms of space, capacity, machine type, and material handling facilities. Each of the five shops is unique. The experience gained from the experimental shop cannot be naturally and easily transferred to others, especially in terms of the design and number of work cells. The developed model was modified to simulate the systems of other shops without excessive effort. Simulating different scenario experiments using a computer can reduce the risks in reconfiguring these shops.

2.8 CONCLUSIONS

A pipe spool fabrication shop is characterized by product uniqueness and by high product mix. These characteristics make the analysis and improvement of the production system very challenging. This research studies applying lean production techniques to shop fabrication, and using a simulation-based approach as a tool to facilitate its implementation. The research demonstrates that one of the lean principles, 'flow', can improve the production performance of pipe spool fabrication shops. The simulation results and real world data analysis concur with the benefits of implementing flow production in industrial construction fabrication. The simulation-based approach used in the study also indicates other potential improvements in a spool fabrication system. This research proves that the developed simulation-based approach is a practical and more powerful tool than the value stream map for modeling and quantitatively evaluating the performance of a complex and dynamic spool fabrication shop. It provides an in-depth understanding of fabrication system performance and the improvements achieved by applying lean techniques. This knowledge may be extended to other areas in construction.

2.9 REFERENCES

Ballard, G. (2000). *Lean Project Delivery System*. Lean Construction Institute.

Barrie, D. S., and Paulson, B. C. (1992). *Professional Construction Management, Including C. M., Design-Construct, and General Contracting*. McGraw-Hill, New York, NY, USA

Hajjar, D., and AbouRizk, S. (2002). "Unified Modeling Methodology for Construction Simulation." *Journal of Construction Engineering and Management*, 128 (2) 174-185.

Halpin, D. W., and Kueckmann, M. (2002). "Lean Construction and Simulation." *Winter Simulation Conference Proceedings*, ed. Enver Yücesan, Chun-Hung Chen, Jane L. Snowdon, and John M. Charnes, SanDiego, California, Vol. 2, pp. 1697-1703.

Howell, G. A. (1999). "What is Lean Construction." *Proceedings Seventh Annual Conference of the International Group for Lean Construction*, IGLC-7, Berkeley, CA, 1-10.

Kambiz, F. (2000). "Using Simulation to Support Implementation of Flexible Manufacturing Cell." *Winter Simulation Conference Proceedings*, ed. Jeffrey A. Joines, Russel R. Barton, Keebom Kang, and Paul A. Fishwick, Orlando, Florida, pp. 1272-1281.

Lian Y-H., and Van L. H. (2002). "An Application of Simulation and Value Stream Mapping in Lean Manufacturing." *Proceedings of the 14th European Simulation Symposium*, ESS2002. Dresden, Germany, SCS.

NSERC/Alberta Construction Industry Research Chair. (2000). "Simphony User's Guide." Edmonton, Alberta, Canada

Tommelein, I. D. (1998). "Pull-driven Scheduling for Pipe-Spool Installation: Simulation of Lean Construction Technique." *Journal of Construction Engineering and Management*, ASCE, 124 (4) 279-288.

Womack, J. P., and Jones D. T. (1996). *Lean Thinking: Banish Waste and Create Wealth in Your Corporation*. Simon & Schuster, New York. 350 p.

CHAPTER 3 – A LARGE SCALE SIMULATION MODELING SYSTEM FOR INDUSTRIAL CONSTRUCTION

3.1 INTRODUCTION

Industrial construction includes a wide range of construction projects essential to our utilities and to basic industries, such as petroleum refineries, petrochemical plants, nuclear power plants, and off-shore oil/gas production facilities (Barrie 1992). The construction phase of industrial construction projects includes piping, civil, structure, electrical works, etc. Piping is always the major and most complicated part of an industrial construction project and is usually located on the critical path of the project schedule. The piping process involves drafting, material procurement and supply, shop fabrication (pipe spools and steel pieces), module assembly in yard, and on-site installation. It is a complex production network system consisting of multiple chains associated with many uncertainties.

To manage the piping process, two management models have been used as underlying theories to generate different strategies (Howell and Ballard 1996). One is the project management (PM) model. It is derived from the activity-centered approach aimed at optimizing the project activity by activity (Howell 1999). "It focuses on individual activities that convert inputs to outputs, conceives them as contractual obligations and spends energy enforcing conformance" (Howell and Ballard 1996). Flaws have been found in the PM framework for managing construction projects. The other management model is the production-based lean construction approach. It considers resources, material flow, and information flow rather than activities. Lean theory roots on the production management mind. “It uses techniques to improve the reliability of those flows and relies on flexibility in proportion to uncontrolled uncertainty. Managing the rate, timing and reliability of handoffs between project participants is at the heart of production management” (Howell and Ballard 1996).

Simulation has been widely used modeling shops and production in the manufacturing industry. It facilitates the implementation of various techniques or theories including lean production. As in the manufacturing industry, simulation can play the same important role when the piping process is managed from the perspective of production. There is another critical issue in such a complex production system: the system cannot optimally improve the overall performance without an understanding of how the processes of subsystems affect the larger system. Therefore, it becomes necessary to model and simulate the whole construction phase of an industrial construction project delivery without losing production level details.

The current construction simulation practice has two important limitations. One limitation concerns the local process modeling required to optimize a sub-system. The other limitation is found in the simplified high level abstract model of the whole system. A local process model is incapable of properly or completely answering the questions beyond the local scope. A simplified abstract project level model likewise fails to capture resource interactions and activity interactions at the production level. There is not a modeling system tailored for the complete industrial construction project system that quantitatively tests different theories to improve system performance.

The special simulation modeling system for industrial construction proposed in this chapter allows the user to develop efficiently a production-based large scale model simulating an entire industrial construction project system. It can capture and reflect features of unique products, resource allocation and interactions, interactions of activities at the production level, and impacts of site conditions. It helps understand the underlying "physics" of production, the effects of dependence, and the variation that emerges along multiple supply chains. It guides us to explore new opportunities to improve system planning performance, instead of taking action only after observing variances between the actual progress and the plan, that is, after problems have already occurred.

Section 3.2 describes the operations performed at different stages of an industrial construction project and the problems existing within industry practice. The developed simulation modeling system for industrial construction is described in detail in Section 3.3. The typical model development using the proposed modeling system and model features are also described in this section. Section 3.4 presents a case study based on a real project undertaken by an Edmonton-based industrial construction contractor. Several system improvement strategies are tested using the developed model.

3.2 BACKGROUND OF INDUSTRIAL CONSTRUCTION

Construction engineers agree that industrial construction is much more complex and uncertain than building construction and infrastructure construction. Industrial construction involves multiple supply chains, which interact with each other. The complexity and most of the uncertainties of industrial construction projects result from the uniqueness characteristics of the product and information. In this research, "product" refers to both mid-product and final product. The final constructed facility is viewed as a product. The mid-products, which refer here primarily to pipe spools, steel pieces, modules, and divisions, are also viewed as products that are handed over by the upstream contractor to the internal or external downstream client. "Information" refers in particular to spool shop drawings, module drawings, and "kanban" information ("Kanban", which is a Japanese word meaning "signal", is a concept in lean production).

A project can be systematically decomposed and represented in a Work Breakdown Structure (WBS). The project is first separated into divisions that represent different physical locations. Each division contains modules and spools. The module is made by assembling spools and equipment on a steel structure. The spools and steel pieces are fabricated in shops. In an industrial construction project, each drafted spool drawing, spool, module, and division is unique. The products and information are quite varied and the products' processes also differ from each other.

The typical process of an industrial construction project is shown in Figure 3-1. Each production phase of an industrial construction project and its existing problems are described below.

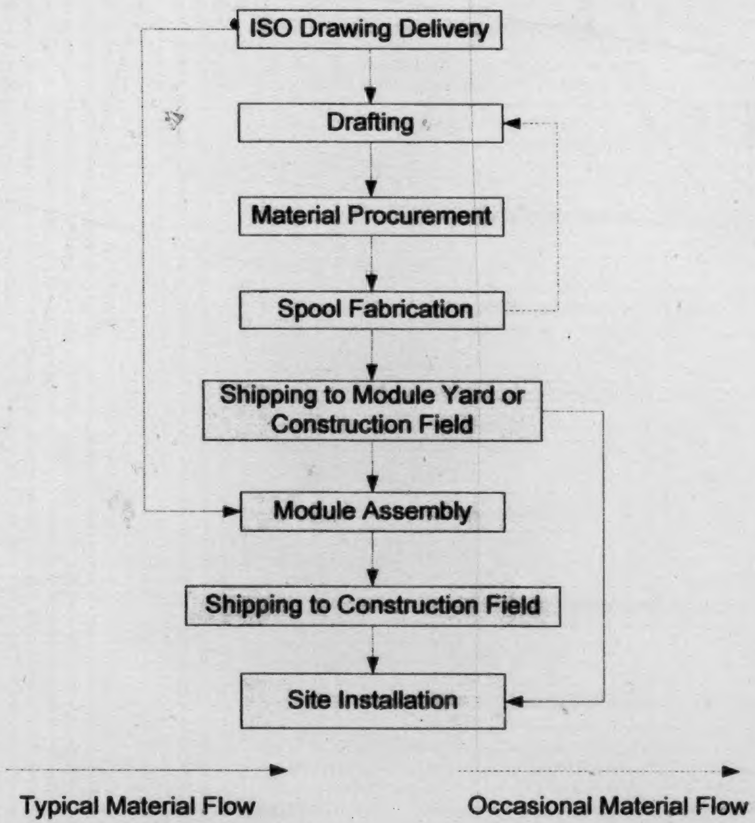

ISO Drawing Delivery
Drafting
Material Procurement
Spool Fabrication
Shipping to Module Yard or Construction Field
Module Assembly
Shipping to Construction Field
Site Installation
Typical Material Flow
Occasional Material Flow

Figure 3-1. Typical Production Process of Industrial Construction

3.2.1 Drafting

The contractor receives the ISO drawings from the client, groups them, and then prioritizes them based on the client's requirements. The draftsperson selects the ISO drawings with the highest priority and completes the drafting and material takeoff tasks, generating spool drawings and the Bill of Material (BOM). The checker then checks the spool drawings and the BOM. The checked drawings go back to the draftsperson who drafted them to be corrected. During redrafting, the draftsperson generates detailed information about welding and labor for the drafted spools. The same checker then checks everything again. Thereafter, a material control person reviews the BOM and fills out the purchase request for material procurement. The spool drawings are stored, waiting to be issued to the fabrication shops once the material is available. Nevertheless it is very common for the drawings to be revised before being issued to the shop. The author's industry survey indicates that the number of shop drawings that were revised could be as high as 62% of the total number of shop drawings for a project. Some drawings were revised as many as six times. The survey data will be shown in Section 3.4.2.

3.2.2 Material Procurement

In industrial construction, materials (pipes and fittings) are partly supplied by clients and partly procured by contractors. When material is delivered, a barcode is generated and pasted onto every piece. The barcode contains information on the job number, control number, and material type. Every piece of material can therefore be tracked throughout the production.

In an effort to control costs, clients occasionally choose to deliver the materials themselves. Contractors complain that the schedule and production suffer when materials are delivered by the clients. Clients do not fully accept that their strategy causes scheduling and production problems for the contractors, which subsequently results in progress delay and other extra costs to themselves. Whether material delivery by clients benefits the overall performance of a project cannot be easily answered.

3.2.3 Spool Fabrication

After the material for a spool is delivered, the shop drawing for the spool can be issued to the shop. A spool normally consists of pipes and fittings (elbow and flange). The typical operations in pipe spool fabrication include cutting, roll-fitting, roll-welding, position-fitting, position-welding, quality control (QC) checking, stress relief, hydro testing, and painting or other surface finishing. Raw pipes are cut to the required size first. They are then roll-fitted, roll-welded, position-fitted, and position-welded. Fabricated spools are checked by the quality crew. Afterwards, a spool may need to undergo any or all of the following operations: stress relief, hydro testing, painting, or other finishing.

The overall performance of the spool fabrication system directly influences the downstream processes of module assembly and site construction. Nevertheless, most spool fabrication shops are inefficiently managed when compared to pure manufacturing shops. This is because every spool is of unique fabrication. This system is also plagued by labor intensiveness, less automation, and frequent change orders from the client after fabrication has started.

Spool fabrication was discussed in detail in Chapter 2. The author used simulation as the tool to facilitate implementation of flow production in a spool fabrication shop to improve the shop's performance.

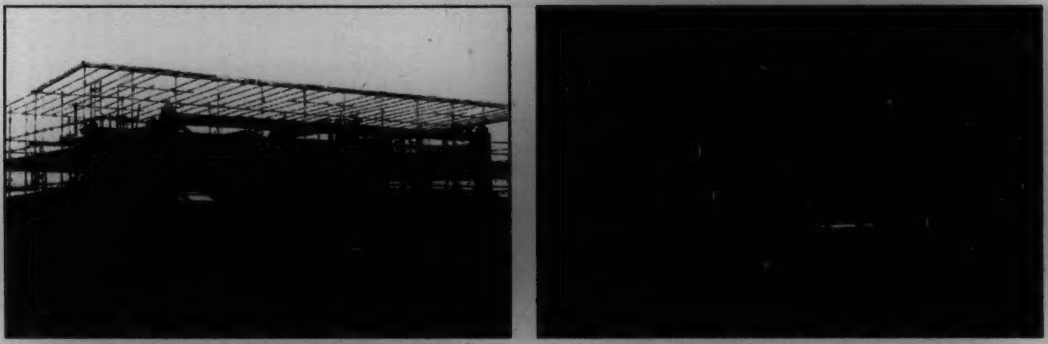
3.2.4 Module Assembly

After fabrication, module spools are shipped to the module assembly yard and non-module spools are directly shipped to the construction site. There are three types of modules: pipe rack module, equipment module, and piping module. A module normally has two to four layers, and is assembled layer by layer. Each layer involves steel erection, scaffolding, and spool installation. Several other operations may be needed and executed following the pipe installation layer by layer: cable tray, light, and instrumentation installation, electrical heat tracing installation, insulation installation, and fire proofing installation. After the last layer is completed, scaffolding is removed. Figure 3-2 is a set of photos illustrating how modules are assembled.

The problem that module assembly often faces is the spool delivery. A module is assembled layer by layer. Each spool is unique and assembled to the designed position of the module. The delivery delay of one spool at the first layer can slow or halt the entire module assembly process, even if all of the spools for the second and third layers have been delivered.

a - Steel erection

b - 1st layer spool installation



c - Scaffolding and spool installation of 2nd and 3rd layer

d - A completed module ready for shipping

Figure 3-2. Module Assembly Process

3.2.5 Site Installation

On-site construction of industrial construction projects is not only about the piping process. There are also building construction, road construction, tower construction, and many other kinds of construction besides the site installation of pipe modules and spools. The piping process, however, is normally on the critical path of the site construction schedule and dominates the progress of an entire project.

The biggest issue related to pipe installation on-site is that the schedule is prone to change. These changes result from uncertainties such as scope changes, site conditions, availability of equipment, or labor resources. These uncertainties are located at the end of the entire supply chain. The resulting changes ripple back to all processes upstream, causing disturbances in module assembly, spool fabrication, material procurement, and the drafting process.

3.3 MODELING SYSTEM DEVELOPMENT

The proposed industrial construction modeling system (ICMS) covers multiple chains and addresses the uniqueness of the product and information flow through industrial construction projects. The modeling system consists of: the product and information modeling component and the production system modeling component. The modeling system architecture is illustrated in Figure 3-3.

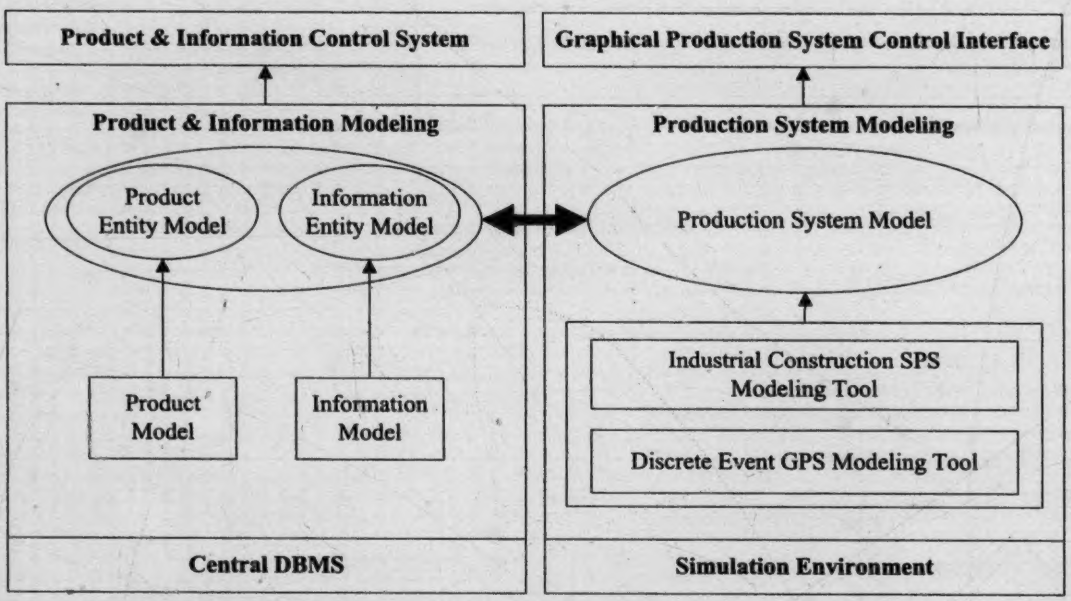

Product & Information Control System
Graphical Production System Control Interface
Product & Information Modeling
Product Entity Model
Information Entity Model
Product Model
Information Model
Central DBMS
Production System Modeling
Production System Model
Industrial Construction SPS Modeling Tool
Discrete Event GPS Modeling Tool
Simulation Environment

Figure 3-3. Industrial Construction Modeling System Architecture

3.3.1 Product and Information Modeling

The product and information modeling mechanism is designed to define the product and information. The defined product or information is viewed as an "entity" from the perspective of the simulation. The system is developed using the relational Database Management System (DBMS).

3.3.1.1 Product Modeling

In an industrial construction project, the facility structure is usually decomposed into smaller components that can be fabricated in the shop or assembled in the yard as described in Section 3.2. In this proposed modeling system, the products that are modeled are raw material, pipe spools, steel pieces, modules, and divisions. The product information carried by the "entity" is classified into Work Breakdown Structure (WBS) information, physical configuration information, and process information.

a. WBS Information

A WBS systematically decomposes a project into measurable elements. A typical WBS used in industrial construction is shown in Figure 3-4. A product's WBS information refers here to its own identity information, any related identity information, and the information of its parent-level product. For example, a spool has a spool number, a control number (drafting drawing number), a module number and layer number of the module into which it will be assembled.

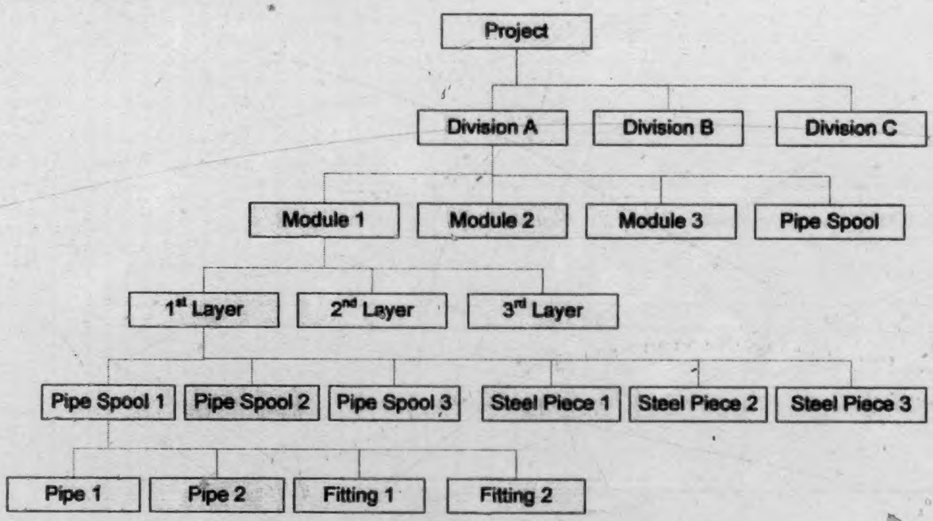

Project
Division A
Division B
Division C
Module 1
Module 2
Module 3
Pipe Spool
1st Layer
2nd Layer
3rd Layer
Pipe Spool 1
Pipe Spool 2
Pipe Spool 3
Steel Piece 1
Steel Piece 2
Steel Piece 3
Pipe 1
Pipe 2
Fitting 1
Fitting 2

Figure 3-4. Work Breakdown Structure

b. Physical Configuration Information

In industrial constructions, each product has its own unique combination of physical configuration data. Modeling an industrial construction system on the production level requires the product's physical information because this information is the dominant factor in its production process, routing, and productivity. Examples of the physical configuration information of a spool are the material type, main diameter inch, length, weight, and welding diameter inch.

c. Process Information

Each product goes through a series of production operations, which are described in detail in Section 3.2. Each product is unique and may require a different combination of operations. The process information defines the process plan for the production of a product. It specifies all required operations and their respective sequence.

3.3.1.2 Information Modeling

Other than product, there is another important type of entity, information, flowing through the industrial construction production system. The main format of the information is the drawing. The information that is modeled in this research includes the spool ISO drawings, spool shop drawings, steel piece drawings, or the module drawings. The information may also include the "kanban" for the pull scheduling technique.

The information carried by the "information entity" is mainly WBS information. A piece of information normally corresponds to a product or a group of products. For example, a module drawing corresponds to a module and a spool shop drawing corresponds to a spool. It should be noted, however, that information flow and product flow are different, separated flows. For example, if the material for a spool is delivered but the drafted shop drawing happens to be revised, the fabrication cannot start.

Information flow is as important as product flow, occasionally more important since information flow causes more problems of delay, rework, or mismatch than material does in many projects. A simulation model without modeling information flow is not a complete model.

3.3.1.3 Data Source

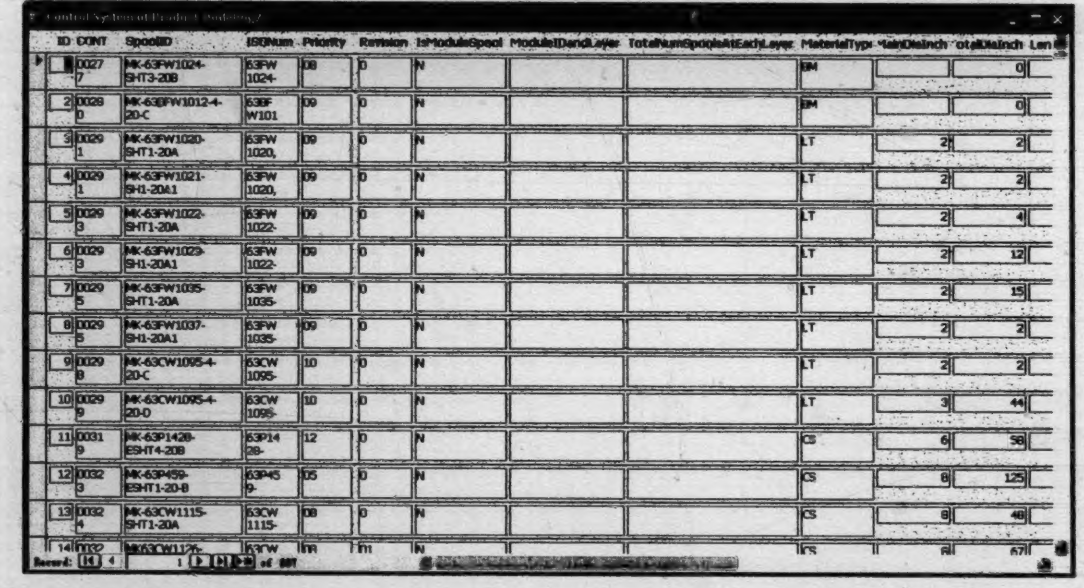
There are a few data sources:

- Computer Aided Design (CAD) tools are widely used in the engineering phase and drafting phase. A computer-generated drawing accommodates a vast amount of product data in an electronic format.
- Information systems are also becoming popular in the construction industry. Companies will use either commercial Enterprise Resource Planner (ERP) systems or their own in-house developed systems. A large amount of information about products and drawings generated during the engineering and drafting phase is stored in the information system. Some information is unavailable in the CAD drawings.
- Computer Numeric Control (CNC) machines, which are widely used in the manufacturing industry, have been used by some creative construction contractors in fabrication shops; however, they have yet to be used popularly in the construction industry. A large amount of production information may be stored in the CNC programs.
- Experience is the last data source when needed data are not available in the above listed data sources. For example, a spool is normally decomposed into a few parts to be fabricated so that each part can be roll-welded. The method of decomposition and the number of parts into which the spool is decomposed are currently decided by experience. Such information is needed to build the simulation model, but is unavailable in any of above systems.

3.3.1.4 Control System

A large amount of product and information data need to be collected and manipulated in the models developed using this modeling system. The system provides interfaces with the CAD system and other information systems such as ERP in order to populate the needed data. It also provides interfaces that allow users to manipulate the stored data. The data are modeled and stored in the central DBMS.

The control system is a set of user interfaces defined to facilitate the management of model information by users. For example, an interface was designed to facilitate the collection and manipulation of spool information, as shown in Figure 3-5. The database and the control system create an open structure for the industrial construction simulation model to interface with a human user and with other existing applications, such as CAD systems, ERP systems, and CNC control programs.



ID	CONT	SpoolID	ISONum	Priority	Revision	IsModuleSpool	ModuleIDandLayer	TotalNumSpoolsAtEachLayer	MaterialTyp	ainDiaInch	otalDiaInch	Len
1	00277	MK-63FW1024-SHT3-20B	63FW1024-	08	0	N			BM		0	
2	00280	MK-63BFW1012-4-20-C	63BFW101	09	0	N			BM		0	
3	00291	MK-63FW1020-SHT1-20A	63FW1020,	09	0	N			LT	2	2	
4	00291	MK-63FW1021-SH1-20A1	63FW1020,	09	0	N			LT	2	2	
5	00293	MK-63FW1022-SHT1-20A	63FW1022-	09	0	N			LT	2	4	
6	00293	MK-63FW1029-SH1-20A1	63FW1022-	09	0	N			LT	2	12	
7	00295	MK-63FW1035-SHT1-20A	63FW1035-	09	0	N			LT	2	15	
8	00295	MK-63FW1037-SH1-20A1	63FW1035-	09	0	N			LT	2	2	
9	00298	MK-63CW1095-4-20-C	63CW1095-	10	0	N			LT	2	2	
10	00299	MK-63CW1095-4-20-D	63CW1095-	10	0	N			LT	3	44	
11	00319	MK-63P1428-ESHT4-20B	63P1428-	12	0	N			CS	6	58	
12	00323	MK-63P459-ESHT1-20-B	63P459-	05	0	N			CS	8	125	
13	00324	MK-63CW1115-SHT1-20A	63CW1115-	08	0	N			CS	8	48	
14	0032	MK63CW1126-	63CW	[illegible]	[illegible]	N			CS	[illegible]	67	

Figure 3-5. Sample User Interface of Control System for Product Modeling

3.3.2 Production System Modeling

The Production System Modeling component enables model developers to build graphically an industrial construction production system. It covers the phases of drafting, material procurement, spool fabrication, module assembly, and site installation. The production modeling system is a combination of a set of customized industrial construction SPS modeling templates, a public module template, and a general-purpose discrete-event simulation tool.

3.3.2.1 Industrial Construction SPS Modeling Templates

A simulation template is a library of basic constructs that model developers can use to simulate a system. The design of the simulation templates of the proposed modeling system employs a Special Purpose Simulation (SPS) approach. SPS enables a practitioner, who is knowledgeable in a particular domain but not a simulation expert, to model a system within that domain using the libraries tailored for that domain (Hajjar and AbouRizk 1999). The developed SPS tools have a high degree of virtual resemblance to the actual systems, which eases the model's development.

The author systematically studied all production processes of industrial construction, resources, material handling systems, space, and decision-making rules of routing. Finally, the modeling elements for industrial construction were abstracted. Because an industrial construction project involves multiple stages or chains, the abstracted and designed library is decomposed into five templates. Each template is composed of a set of the abstracted modeling elements for a specific stage of production: drafting, material procurement, spool fabrication, module assembly, and site installation. The *Simphony* modeling environment is chosen to implement these designed templates. *Simphony* is a simulation platform for building SPS templates and models. It allows users to implement highly flexible simulation tools supporting graphical, hierarchical, modular, and integrated modeling (Hajjar and AbouRizk 2002). The description of the developed modeling elements and their graphic notations are given in Table 3-1.

Table 3-1. SPS Modeling Elements for Industrial Construction

ELEMENT	NOTATION	DESCRIPTION
DRAFTING TEMPLATE		
DraftSIM	N	Represents the whole drafting department.
Draft	i	The draftsperson drafts a package of drawings.
Check	✓	The checker checks a package of drawings.
ResourceTracer	👣	Track the draftsperson or checker who drafts or checks the drawing for each drawing package.
ResourceRouter	IF→E	Dispatch the drawings packages back to the draftsperson or checker who drafted or checked them during the process of back drafting or back checking.
BackDraft	i✓	The package of drawings is back-drafted by the draftsperson who drafted it during drafting process.
BackCheck	✓	The package of drawings is back-checked by the checker who checked it during checking process.
MATERIAL PROCUREMENT TEMPLATE		

ProcurementSIM		Represents the whole material procurement process including material delivery by owner.
Procurement&-Delivery		Represents the whole duration of material procurement and delivery.
SPOOL FABRICATION TEMPLATE		
FabPlant		Represents a fabrication plant. It is the parent of all shop elements.
Shop		Represents a shop. It can be a fabrication bay, a stress relief shop, or a paint shop.
Workcell		Represents a work cell where a spool is fitted and welded. It has a certain capacity, which only allows user defined number of products or diameter inches to be processed. It also can be a cutting station or QC checking station.
LaydownArea		Represents a buffer area where products can stay and wait for next process. It has a certain capacity, which only allows a user defined number of products to stay.
ProcessController		Simulates a decision maker to decide if the spool goes to next standard process or not.
DispatchController		Simulates a decision maker to dispatch spools to one of the destinations that have same process

		function (e.g. Bays, or WorkCells) based on certain criteria.
CapacityDetector		Simulates a decision maker detecting the current capacity status remaining at the next destination (e.g. a Bay, a WorkCell, or a LayDown.) of the product flow to determine if the product can go to this destination or not.
Resource		Represents a container of any equipment or laborer. It can model various interruptions.
Path	━	Defines the material handling system inside and outside of the shop. It represents a route that products are handled through from a location to another location. The duration of the movement can be defined.
DrawingTool *(3 implicit elements)*	N.A.	Can create layout gridlines and can import plant and shop layout drawings.
MODULE TEMPLATE		
ModuleSIM	▦	Represents the whole module assembly yard.
1stLayer	Ist	Represents a container for all production activities at the 1st layer of a module.
2ndLayer	2nd	Represents a container for all production activities

		at the 2nd layer of a module.
3^{rd}Layer	3rd	Represents a container for all production activities at the 3rd layer of a module.
RemainingWork	▨	Represents a container for all remaining activities after spool installation of the last layer of a module.
SITE INSTALLATION TEMPLATE		
SiteSIM	Site	Represents the whole construction site.

3.3.2.2 Public Simulation Modules for Industrial Construction

There are nine common elements designed and useful in any of the above five SPS templates. They are implemented in *Simphony* as well. The description of these modeling elements and their graphic notations are given in Table 3-2.

Table 3-2. Public Module Elements for Industrial Construction

ELEMENT	ICON	DESCRIPTION
DatabaseImporter	213→	Imports products or information defined by the Product & Information model in the central database to the Production System model.
TimeCollector	⏱	Records In time and Out time of entity at any location (Workcell, Laydown Area). They have to be used by pair in any location.
DatabaseExporter	→798	Exports all information collected for products from

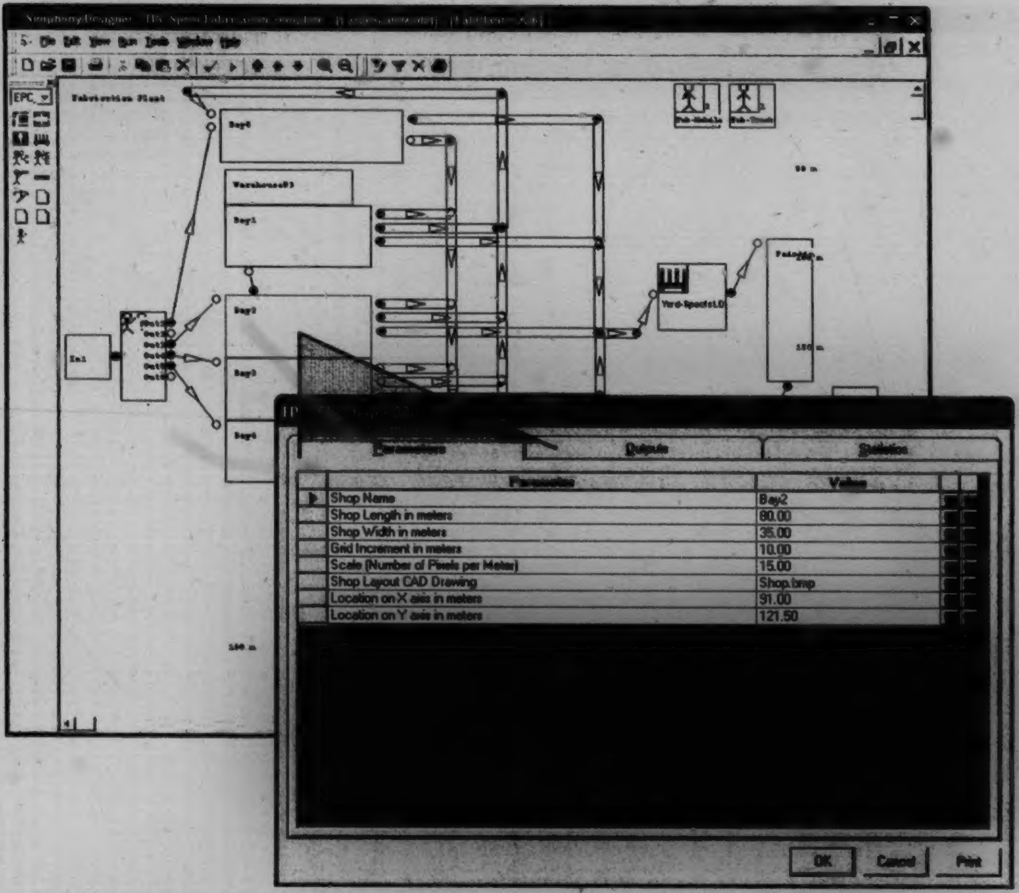
		Production System model back to the central database of Product & Information model.
Batch		These two elements are always coupled for use. A predefined number of entities can be batched by a *Batch* element without being changed during batched period. They can be released to original state by an *Unbatch* element without losing any information on any entity.
Unbatch		
Assembly		This is an intelligent element. It can search and draw all components with same ID from a queue and merge them into a product and send it out.
SCMatching	■	Performs the function of matching upstream material, pre-fabricated parts, and information for downstream. It works similarly to the *Assembly* element.
KanbanSender		These two elements are always coupled for use. A *KanbanSender* element sends signal, which is normally an attribute value of the passing entity, from downstream to upstream to pull required material (drawings, spools, or modules). A *KanbanReceiver* element receives the signals sent by a *KanbanSender* element and makes response by adjusting the priority of passing entities at upstream.
KanbanReceiver		

3.3.2.3 General-Purpose Discrete-Event Simulation Tool

Simphony's Common Template (AbouRizk and Mohamed 2000) is utilized as the General-Purpose Simulation (GPS) tool. The Common Template provides most of the required basic modeling elements for general-purpose discrete-event simulation. In the proposed simulation system, the GPS tool is used by model developers to simulate fundamental processes at the micro level, while cooperating with the above SPS modeling templates. It helps extend the capability and flexibility of the above SPS modeling tools. The basic functionality of the most important modeling elements in the Common Template was summarized in Table 2-1 in Chapter 2.

3.3.2.4 Control Interface

The graphic control for the production modeling system is provided through *Simphony* and customized during the development of the above SPS modeling elements. Example graphic interfaces to develop the models and to manipulate data are shown in Figure 3-6.


Fabrication Plant
Bay0
Warehouse03
Bay1
Bay2
Bay3
Bay4
In1
Yard-Spaces
150 m
100 m
Parameters
Outputs
Statistics
OK
Cancel
Print

	Parameter	Value
▶	Shop Name	Bay2
	Shop Length in meters	80.00
	Shop Width in meters	35.00
	Grid Increment in meters	10.00
	Scale (Number of Pixels per Meter)	15.00
	Shop Layout CAD Drawing	Shop.bmp
	Location on X axis in meters	91.00
	Location on Y axis in meters	121.50

Figure 3-6. Sample User Interface of Control System for Production System

3.3.3 Typical Model Development and Features

Model developers can use any of the developed SPS templates, public modules, or GPS tool to model a particular phase of industrial construction, or they can use all of them to model the entire production system; this research explores for the latter approach. Several important features of a typical model, important development procedures, and interactions among modules are explained and highlighted below.

The *DatabaseImporter* element imports drawing entities defined in the product and information model to the production system model. The drawing entities are batched by the *Batch* element and sent to the draftsperson and checker for drafting and checking. The *ResourceTracer* element remembers who drafted or checked the drawing. The *ResourceRouter* element then locates the same person to do the backdraft and the backcheck. After material checking, the *Unbatch* element releases the grouped drawings to their original state. Figure 3-7 depicts the flow diagram illustrating the described process.

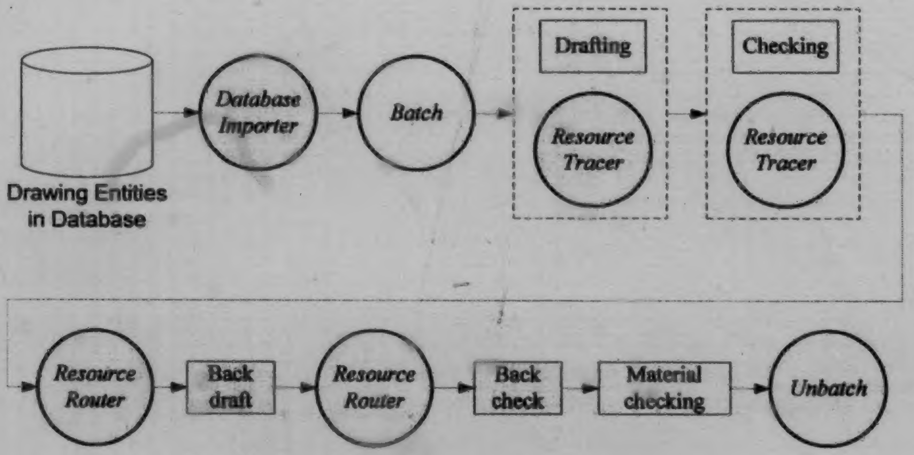

Drawing Entities in Database
Database Importer
Batch
Drafting
Resource Tracer
Checking
Resource Tracer
Resource Router
Back draft
Resource Router
Back check
Material checking
Unbatch

Figure 3-7. Modeling Flow Diagram (a)

Thereafter, the entity flow follows two paths: material flow and drawing flow. The material flow goes through the *ProcurementSIM* element. The drawing flow has to wait for delivery of material before going to the shop; however, a certain percentage of the drawings could be revised during this waiting period. Figure 3-8 depicts the flow diagram illustrating the described process.

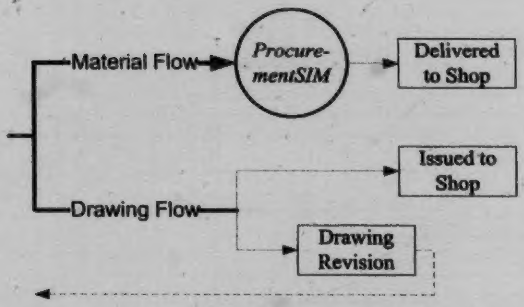
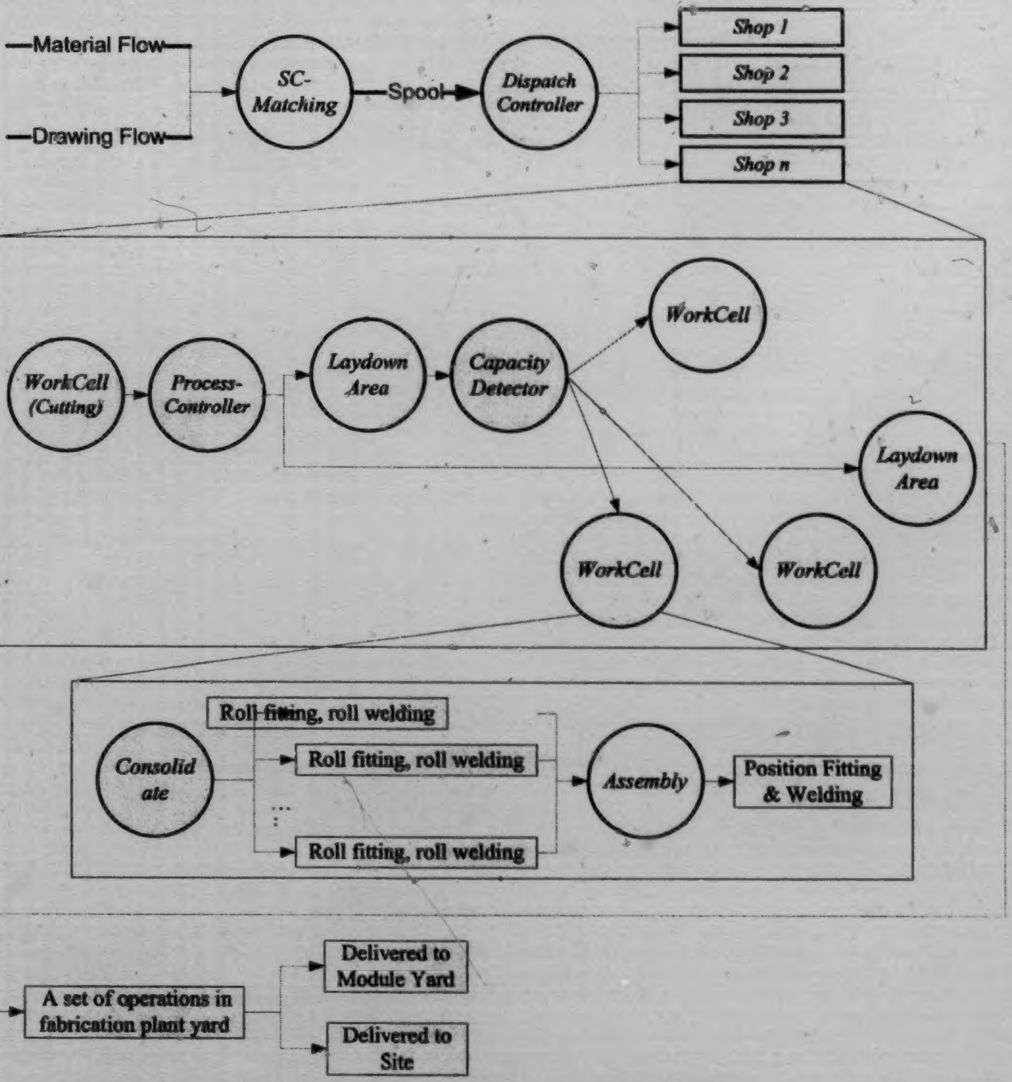


Figure 3-8. Modeling Flow Diagram (b)

The drawing and materials for one spool meet, match, and become one spool product entity through the *SCMatching* element, which initiates the fabrication process. The entity is then dispatched to one of the fabrication shops by the *DispatchController* element based on its configuration information. The product entity always travels first into a *ProcessController* element to determine the next operation, based on the product's process information. Cutting is always the first operation in a fabrication shop. The product then goes to a *LaydownArea* or to the next *WorkCell*; before going to either of these elements, however, the product must travel through a *CapacityDetector* element to detect if the destination has enough capacity left to accommodate it or not. If it does, the entity will travel through the material handling system, which will direct it to the destination; otherwise the product has to wait at its current location. During fabrication, a spool product entity is normally decomposed into a few parts by a *Consolidate* element and processed by a series of operations, and then finally assembled by an *Assembly* element. Products might be batched by the *Batch* element and unbatched by the *Unbatch* element for handling. The *Conditional* element sends the fabricated spool entities to the

ModuleSIM element or to the *SiteSIM* element based on if the spool is a module spool or not. Figure 3-9 depicts the flow diagram illustrating the described process.



Material Flow
Drawing Flow
SC-Matching
Spool
Dispatch Controller
Shop 1
Shop 2
Shop 3
Shop n
WorkCell (Cutting)
Process-Controller
Laydown Area
Capacity Detector
WorkCell
Laydown Area
WorkCell
WorkCell
Consolidate
Roll fitting, roll welding
Roll fitting, roll welding
...
Roll fitting, roll welding
Assembly
Position Fitting & Welding
A set of operations in fabrication plant yard
Delivered to Module Yard
Delivered to Site

Figure 3-9. Modeling Flow Diagram (c)

The module drawing, steel pieces, and all the spools for the first layer of the module meet at an *Assembly* element and merge into one conceptual module product entity, triggering the start of the first layer assembly of the module in the *1stLayer* element.

Thereafter, the entity representing the module goes through the production system layer by layer. It meets the arriving spools for the second layer of the module at an *Assembly* element and they merge into one entity to trigger the start of the second layer assembly of the module in the *2ndLayer* element. This continues at the *3rdLayer* element and the *4thLayer* element until the completion of the module. Figure 3-10 depicts the flow diagram illustrating the described process. The site installation drawing, spools, and modules for a division meet and match at the *SiteSIM* element to trigger the start of site installation.

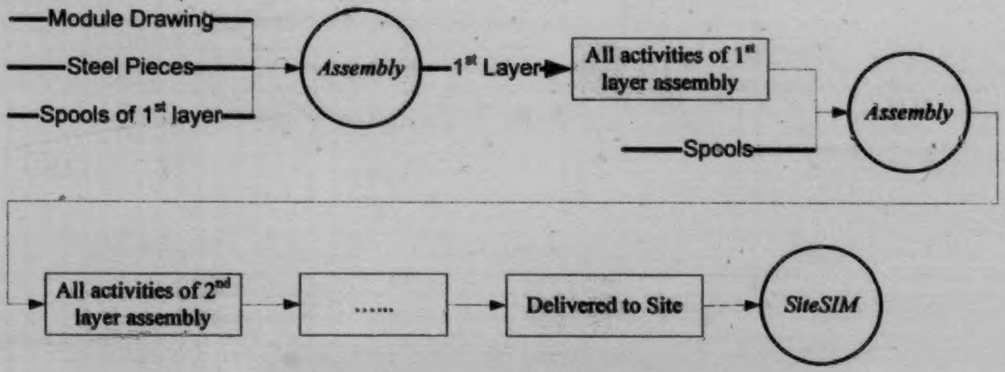

Module Drawing
Steel Pieces
Spools of 1st layer
Assembly
1st Layer
All activities of 1st layer assembly
Spools
Assembly
All activities of 2nd layer assembly
……
Delivered to Site
SiteSIM

Figure 3-10. Modeling Flow Diagram (d)

The *Time Collector* element can collect specific information defined by model developers during modeling. Upon the completion of a simulation experiment, the *Database Exporter* element exports predefined collected data to the central DBMS for reports and analysis by users.

3.4 CASE STUDY

3.4.1 Case Description

A project executed by an Edmonton-based industrial construction contractor was selected as a case study. Figure 3-11 illustrates the basic flow diagram of this project. 687 shop drawings are drafted based on the priorities established in the project. They consist of 475 spool drawings, 187 module material drawings, and 25 non-module material drawings. During drafting stage, however, all of them are usually called spool shop drawing. Materials are procured based on the bill of material (BOM) information provided at the drafting stage.

The 475 shop drawings were then issued to fabrication shops batch by batch, and materials for the 475 spools were delivered to the fabrication shops. The 475 spools were scheduled for fabrication at three out of the five shops operated by the contractor. They were split up based on the material involved and the size of the spool. The 187 pieces of module material are delivered to the yard. 25 pieces of non-module material are delivered to the construction site. Of the fabricated spools, 437 are module spools and are delivered to the module yard, and 38 are non-module spools and are shipped to construction site. The 437 module spools, 187 pieces of module materials, and the required steel pieces are then assembled into 11 modules at the yard. The 11 assembled modules, non-module spools, and non-module materials are delivered to the construction site and installed into three divisions.

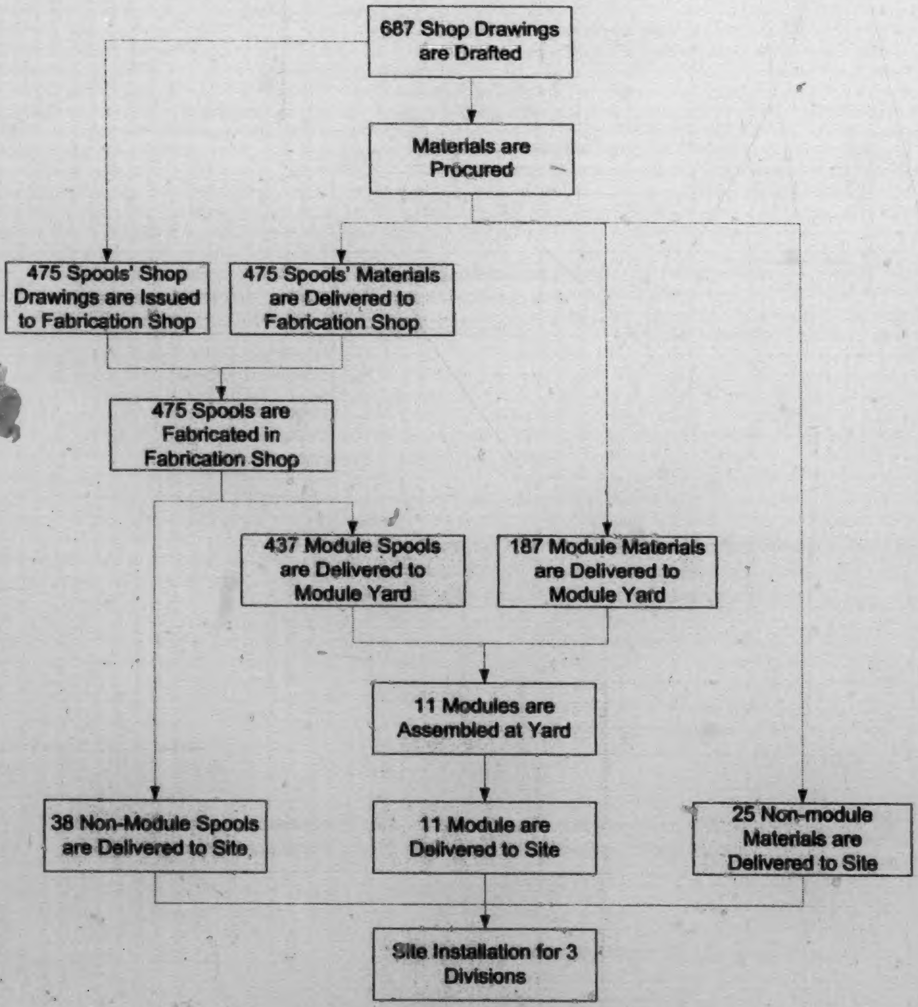

687 Shop Drawings are Drafted
Materials are Procured
475 Spools' Shop Drawings are Issued to Fabrication Shop
475 Spools' Materials are Delivered to Fabrication Shop
475 Spools are Fabricated in Fabrication Shop
437 Module Spools are Delivered to Module Yard
187 Module Materials are Delivered to Module Yard
11 Modules are Assembled at Yard
38 Non-Module Spools are Delivered to Site
11 Module are Delivered to Site
25 Non-module Materials are Delivered to Site
Site Installation for 3 Divisions

Figure 3-11. Project of Case Study

The simulation model under development will be driven by information on the entities of drawings, spools, and modules. Entities are defined as carrying a large amount of information. Table 3-3 lists the associated properties of these entities.

Table 3-3. Properties of Entities

Drawing	Spool	Module properties
ControlNumber	*ControlNumber*	*DevisionID*
ISONumber	*SpoolID*	*ModuleID*
Priority	*Priority*	*ModuleIDandLayer*
Revision	*IsModuleSpool*	*TotalNumLayersOfEachModule*
	DevisionID	*TotalNumSpoolsAtEachLayer*
	ModuleID	
	ModuleIDandLayer	
	TotalNumLayersOfEachModule	
	TotalNumSpoolsAtEachLayer	
	MaterialType	
	MainDiaInch	
	TotalDiaInch	
	Length	
	Weight	
	PartsQuantity	
	NeedsCutting	
	NeedsFittingWelding	
	NeedsRT	
	NeedsLT	
	NeedsMT	
	NeedsHT	
	NeedsStressRelief	
	NeedsHydroTest	
	NeedsPainting	
	Schedule	

The contractor's fabrication plant has five fabrication shops, one stress relief shop, and one painting shop. The lay down areas are distributed throughout the plant. The second layer of the simulation model will demonstrate virtually the fabrication plant configuration. The third layer of the simulation model will demonstrate virtually the configuration of the fabrication shops. The module assembly yard is adjacent the fabrication plant.

The industrial construction project is a large and complex production system. Managing such a system faces challenges such as an extreme product mix and uncontrolled uncertainties. Strategies and techniques have been proposed by researchers to help improve the production system (Howell and Ballard 1996, Tommelein 1998). The proposed modeling system is used to model the entire production system of the contractor based on this case project. The model provides a laboratory for testing the various strategies proactively and quantitatively. Several experiments testing the different strategies were conducted and are presented in Sections 3.4.3 to 3.4.6.

3.4.2 Model Development

A pilot model was built for this project. The product and information model was populated mainly by importing data from CAD drawings of the project and the contractor's in-house developed information system. A small part of the information was manually collected and entered by the author. The entire production system described in Section 3.2 was digitally mapped onto the production model, being tailored to the system configuration of the contractor. Figure 3-12 depicts the product and information model, the first level of the production system model hierarchy, and their relationship. The production system is shown by the screenshot of the developed virtual model. It integrates all the subsystems at the highest level and communicates with the product and information model.

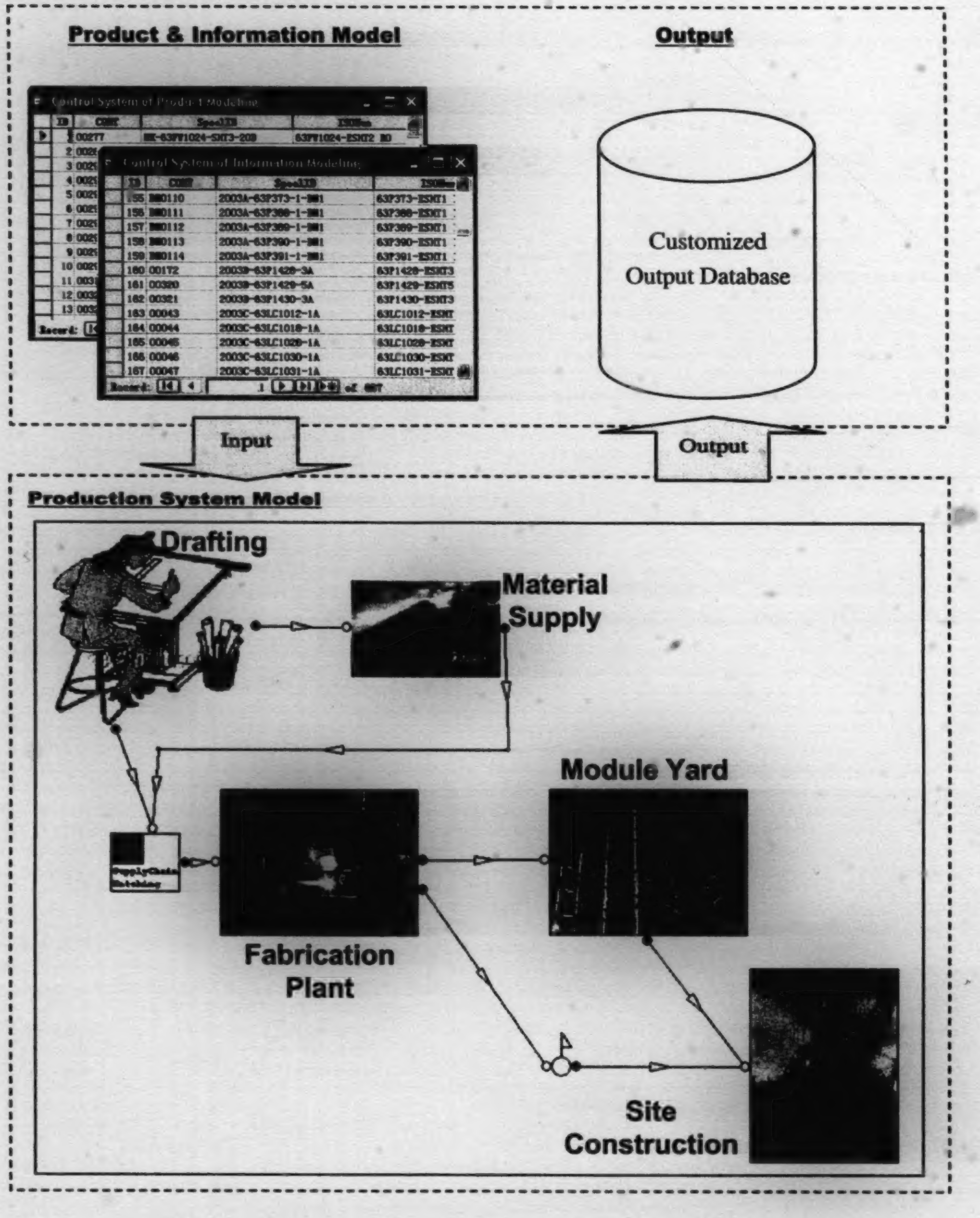

Product & Information Model
Output
Control System of Product Modeling
Control System of Information Modeling
Customized
Output Database
Input
Output
Production System Model
Drafting
Material
Supply
Module Yard
SupplyChain
Matching
Fabrication
Plant
Site
Construction

Figure 3-12. First Level of the Developed Model

Figures 3-13 to 3-17 illustrate the subsystems using screenshots of the virtual model in a hierarchical structure.

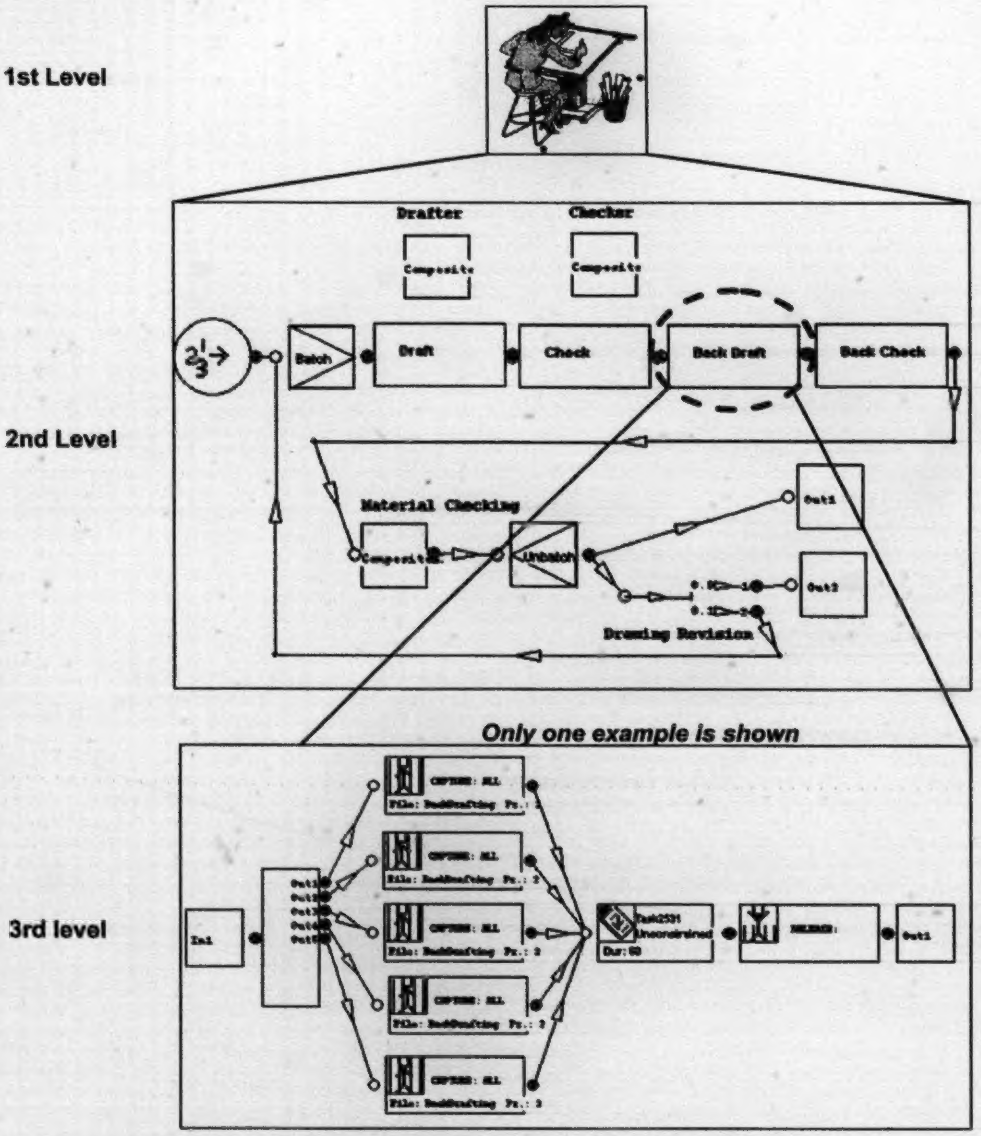

1st Level
2nd Level
3rd level
Drafter
Checker
Composite
Composite
Batch
Draft
Check
Back Draft
Back Check
Material Checking
Composite
Unbatch
Out1
Out2
Drawing Revision
Only one example is shown
In1
Out1
Out2
Out3
Out4
Out5
CAPTURE: ALL
File: BackDrafting
Task2531
Unconstrained
Dur:60
RELEASE:
Out1

Figure 3-13. Display of Hierarchical Model of Drafting System

In Figure 3-13, the second level shows the full model of the drafting system. An example of a detailed process model for back drafting is illustrated in the third level.

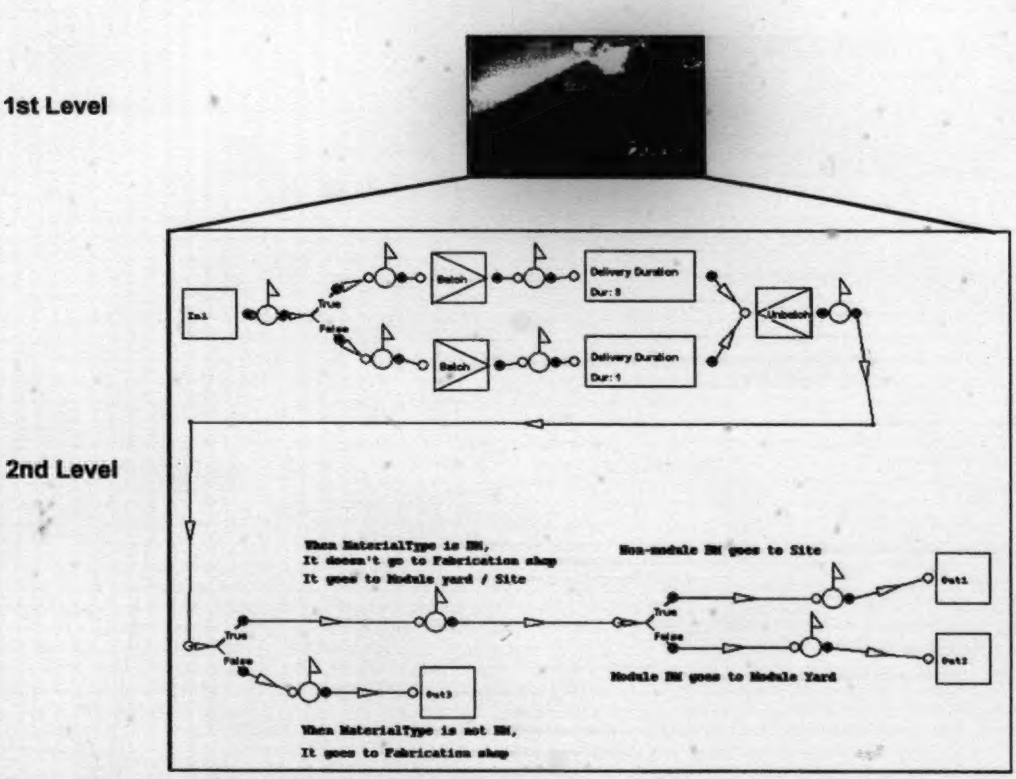

1st Level
2nd Level
Ini
True
False
Batch
Batch
Delivery Duration
Dur: 3
Delivery Duration
Dur: 1
Unbatch
When MaterialType is BM,
It doesn't go to Fabrication shop
It goes to Module yard / Site
Non-module BM goes to Site
True
False
True
False
Out1
Out2
Out3
Module BM goes to Module Yard
When MaterialType is not BM,
It goes to Fabrication shop

Figure 3-14. Display of Hierarchical Model of Material Procurement

The material procurement model has only two hierarchical layers. In Figure 3-14, the second level shows the full model of the material procurement system.

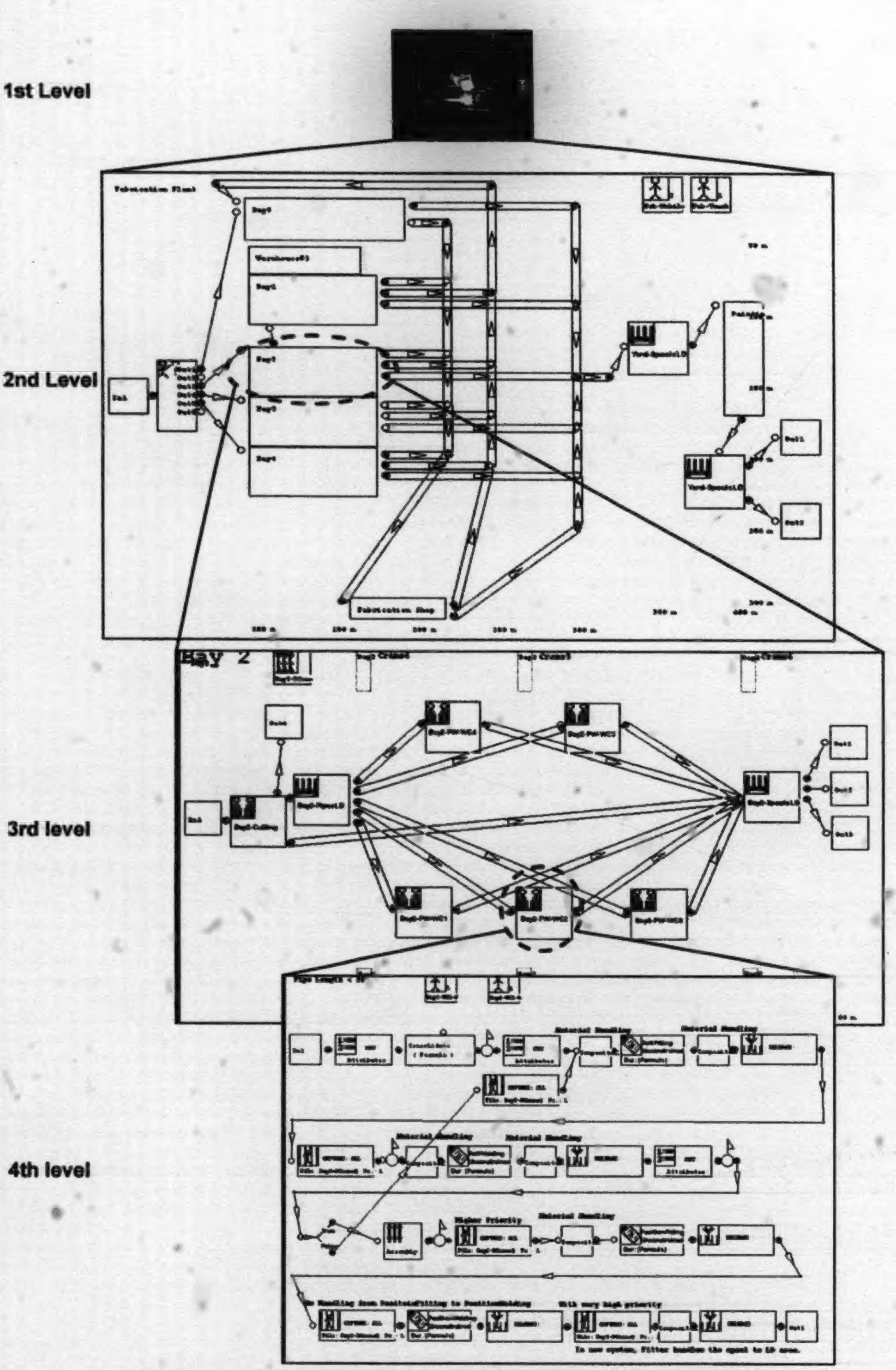

1st Level
2nd Level
3rd level
4th level
Bay 2
Fabrication Shop
Material Handling

Figure 3-15. Display of Hierarchical Model of Fabrication

Figure 3-15 is the hierarchical model of the fabrication system. The second level shows virtually the entire fabrication plant. It illustrates the layout of the plant. An example of Shop 2 is illustrated in the third level. It shows the configuration of a work cell-based spool fabrication shop. The fourth level is a detailed process model for one of the work cells.

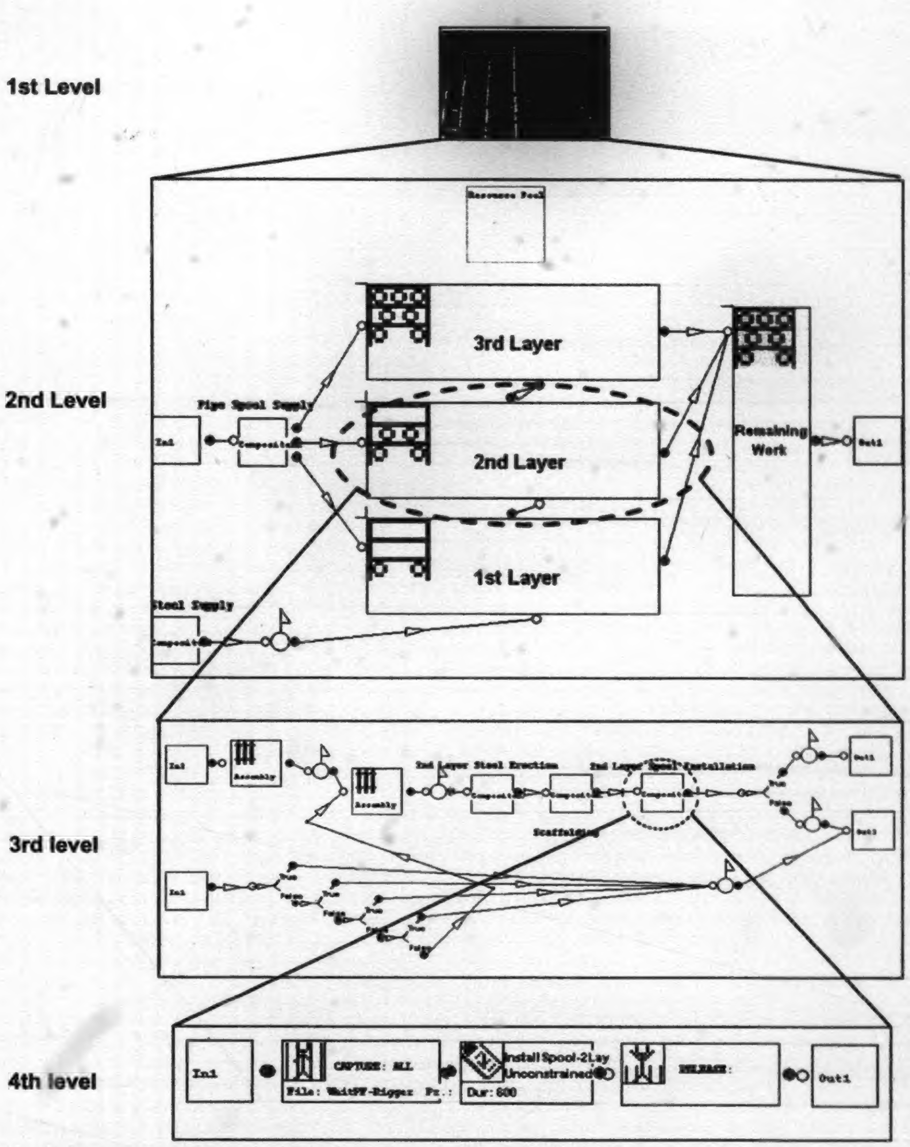

1st Level
2nd Level
3rd level
4th level
Resource Pool
3rd Layer
2nd Layer
1st Layer
Remaining Work
Pipe Spool Supply
Steel Supply
In1
Out1
Composite
Assembly
2nd Layer Steel Erection
2nd Layer Spool Installation
Scaffolding
True
False
CAPTURE: ALL
File: WaitPF-Rigger
Install Spool-2Lay Unconstrained
Dur: 800
RELEASE:

Figure 3-16. Display of Hierarchical Model of Module Assembly

Figure 3-16 is the hierarchical model of the module assembly system. The second level shows the whole process of module assembly. The third level is the process model depicting the second layer's assembly. Some processes will require the fourth level be detailed.

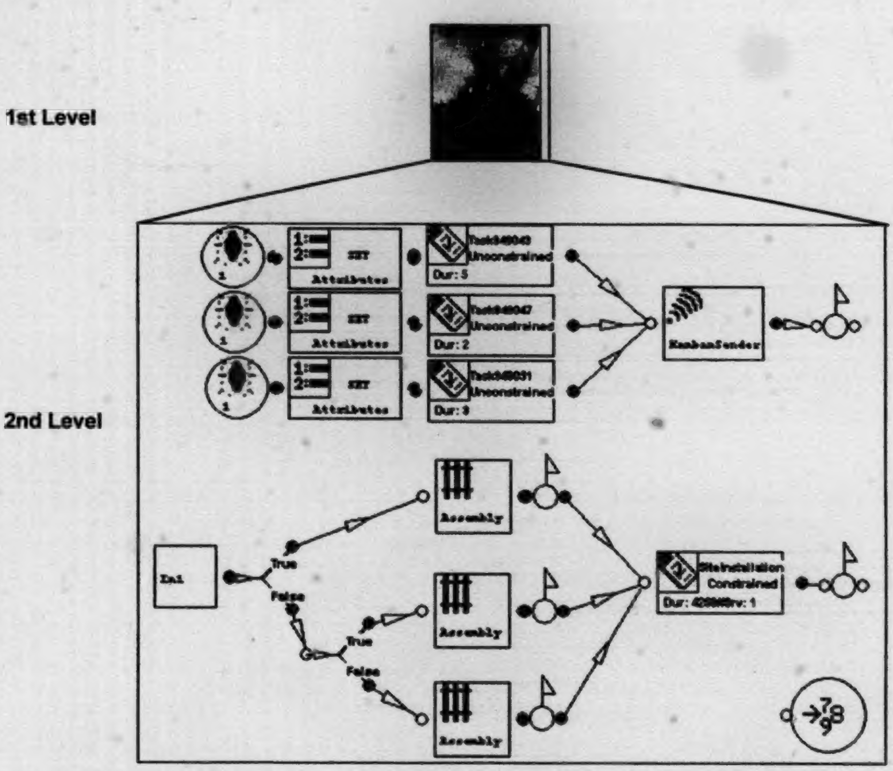

1st Level
2nd Level
SET
Attributes
Unconstrained
Dur: 5
Dur: 2
Dur: 3
KanbanSender
Ini
True
False
Assembly
SiteInstallation
Constrained

Figure 3-17. Display of Hierarchical Model of Site Installation

Figure 3-17 is the hierarchical model of the pipe installation at the construction site. The second level only shows part of the processes of site installation. The *DatabaseExportor* element exports the collected data to the central database. The construction site's system was not modeled in sufficient detail due to the time constraints.

3.4.3 Impacts of Drawing Revisions

Most construction simulation models do not model information flow. In industrial construction, information flow is separate from product flow. The drafting department is also a production system and the drawings are processed by certain resources. Without the arrival of information, the fabrication cannot start even if the material has been delivered. The issue of drawing revision has recently become more significant in industrial construction in Alberta because of how the projects' deliveries are compacted by the owners. Drawing revisions result from changes to the engineering made by the engineering firms or due to the contractor's drafting quality. Drawing revision causes a lot of interruptions or delays in the production system, increases cycle time, and results in additional costs for the project. It has been one of the main reasons for progress delay in many industrial construction projects.

The author conducted interviews with the drafting supervisor and several fabrication project coordinators working with the contractors. The proportion of revisions to shop drawings conducted during the drafting stage is unexpectedly high. Of all the fabrication projects completed by this contractor in Alberta during the last five years, the best engineering firm could control the drawing revision percentage to 10%, while the worst case is 35%. One of the main reasons for such a high number revisions is that more projects have become fast-track projects. A statistical analysis of drawing revisions was conducted on the spool shop drawings for three finished projects by this contractor in Edmonton. Certain shop drawings were revised up to five or six times. In the statistical analysis, drawings which were revised 0, 1, 2, 3, 4, 5, and 6 times were counted. The percentage they represented of the total number of drawings was calculated as well. The statistical analysis results are shown in Table 3-4. In this analysis, the drawing revisions were caused either by the engineering firm or by the contractor. The cause of each revision is not traceable using the contractor's tracking system. The drawing control person working on these three jobs assessed that approximately 60% of the revisions were caused by the engineering firm.

Table 3-4. Drawing Revision Analysis

[redacted]

Revision Times	Number of shop drawings	Percentage	
0	71	**37.97%**	
1	38	20.32%	**62.03%**
2	43	22.99%	
3	27	14.44%	
4	7	3.74%	
5	1	0.53%	
Total number of drawings:	**187**	**100.00%**	

[redacted]

Revision Times	Number of shop drawings	Percentage	
0	67	**46.85%**	
1	62	43.36%	**53.15%**
2	9	6.29%	
3	3	2.10%	
4	2	1.40%	
Total number of drawings:	**143**	**100.00%**	

[redacted]

Revision Times	Number of shop drawings	Percentage	
0	2500	**72.51%**	
1	775	22.48%	**27.49%**
2	140	4.06%	
3	24	0.70%	
4	5	0.15%	
5	2	0.06%	
6	2	0.06%	
Total number of drawings:	**3448**	**100.00%**	

The effect of drawing revision on the performance of an entire project is an important area requiring study. The sensitivity analysis of drawing revision is conducted based on the developed model. The drawing revision is modeled after material checking is accomplished, as shown in Figure 3-18. The probability that revision will take place is not related to the percentage of revised drawings out of the total number of drawings, but will instead occur regardless of whether the drawing has already been revised or not. As shown in the above data, taken from the industry, a certain percentage of drawings were revised more than once. In the case study, the maximum revision times for a drawing are set at three times.

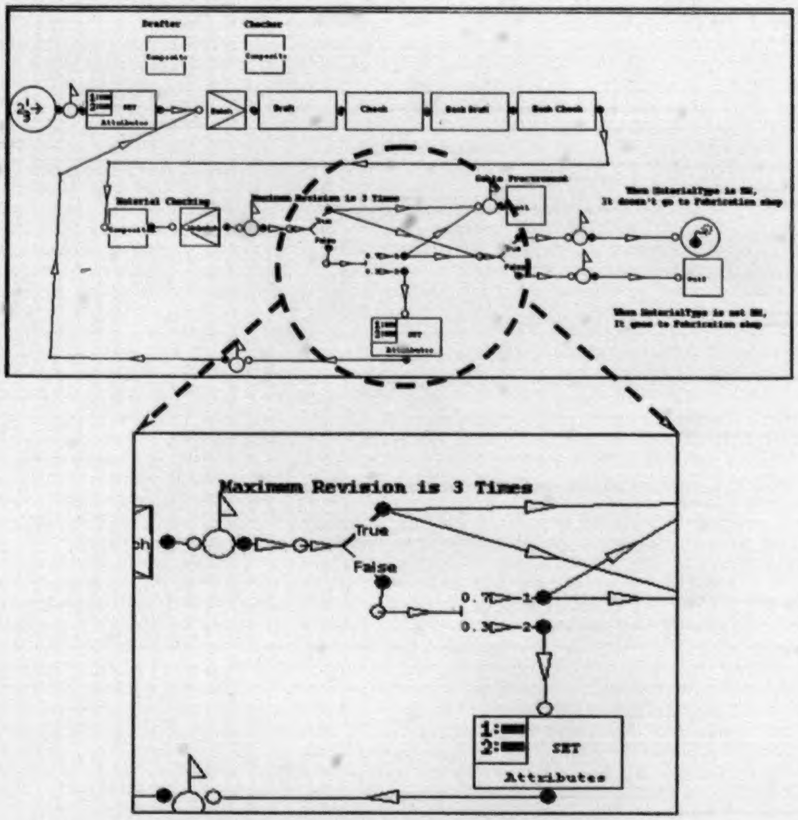

Drafter
Checker
Draft
Check
Back Draft
Back Check
Material Checking
Maximum Revision is 3 Times
True
False
0.7
1
0.3
2
1:
2:
SET
Attributes

Figure 3-18. Modeling Drawing Revision

The probability that a drawing is revised can be reduced through experimentation. Five experiments were executed to test the impact of drawing revision on the project delivery time. These experiments generated probabilities of revision at 0%, 10%, 20%, 30%, and 40%, respectively. The result is shown in Figure 3-19. The simulation results indicate that the project progress is sensitive to drawing revision. Strategies should be taken to reduce drawing revision.

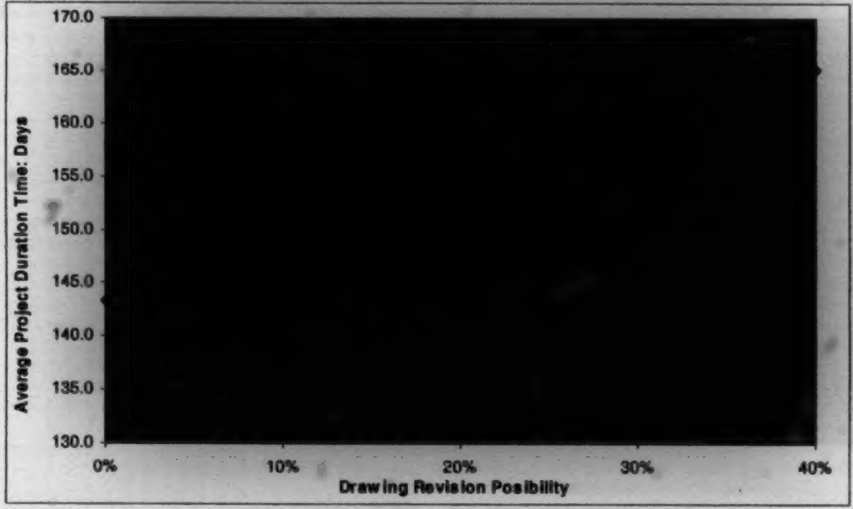


Figure 3-19. Sensitivity Analysis of Project Duration to Drawing Revision

3.4.4 Effects of Different Material Delivery Strategies

As discussed in Section 3.2.2, both the contractors' schedules and the production itself suffer due to material delays, which subsequently result in progress delay and additional costs for all stakeholders. The contractor involved in this research is willing to use the developed model as a tool in order to convince its clients to change their current material supply strategy because the contractor believes the project progress will be performed better and faster if it can procure and supply materials itself. The clients or engineering firms, however, believe they can control the cost better if they control the material procurement and delivery. The overall performance of the project using these different material delivery strategies is difficult to evaluate. It would require a comprehensive analysis including cost, finance, and quality.

The developed model can be used to experiment with different material delivery strategies. These cases are hypothetical situations suggested in order to demonstrate the capability of the built simulation model. Three scenarios are described below:

- Scenario 1: Delays in delivery and mismatches of the material delivered happen randomly to 30% of the spools in the project. The receipts of materials for these 30% of spools are assumed to be delayed by 10 days. The modeling is shown in Figure 3-20.

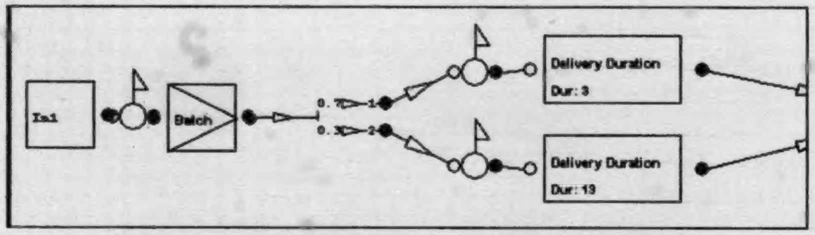


Figure 3-20. Modeling Material Delivery Strategy 1

- Scenario 2: The materials for the first 340 spools (approximately 50%) out of the total 687 spools are delayed or delivered in the wrong type or size. Delays and mismatches of the delivery randomly happen to 30% of these spools. Their delivery is assumed to be delayed by 10 days. The materials for the remaining 347 spools (approximately 50%) are supplied with no delays or mistakes. The modeling is shown in Figure 3-21.

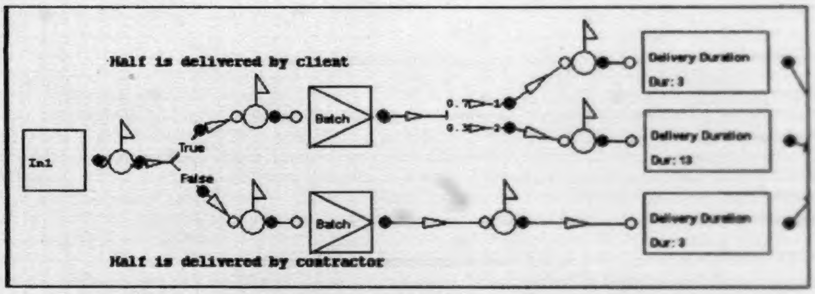

Half is delivered by client
Ini
True
False
Batch
0.7
1
0.3
2
Delivery Duration
Dur: 3
Delivery Duration
Dur: 13
Batch
Delivery Duration
Dur: 3
Half is delivered by contractor

Figure 3-21. Modeling Material Delivery Strategy 2

- Scenario 3: Materials for the project are purchased and delivered without any delay or mismatch. The modeling is shown in Figure 3-22.

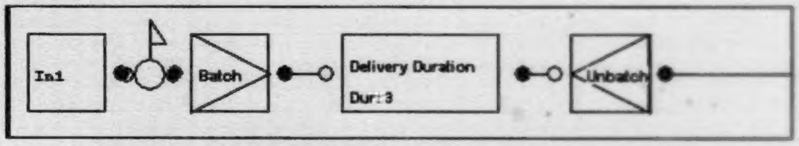

Ini
Batch
Delivery Duration
Dur: 3
Unbatch

Figure 3-22. Modeling Material Delivery Strategy 3

The results are shown in Figure 3-23. It indicates that Strategy 3 is the best option if the project performance is evaluated by the project progress. It implies that the project duration is sensitive to material delays. The client, the engineering firm, and the contractor should therefore pay attention in order to select the material delivery strategy most effective in eliminating material delay and mismatch.

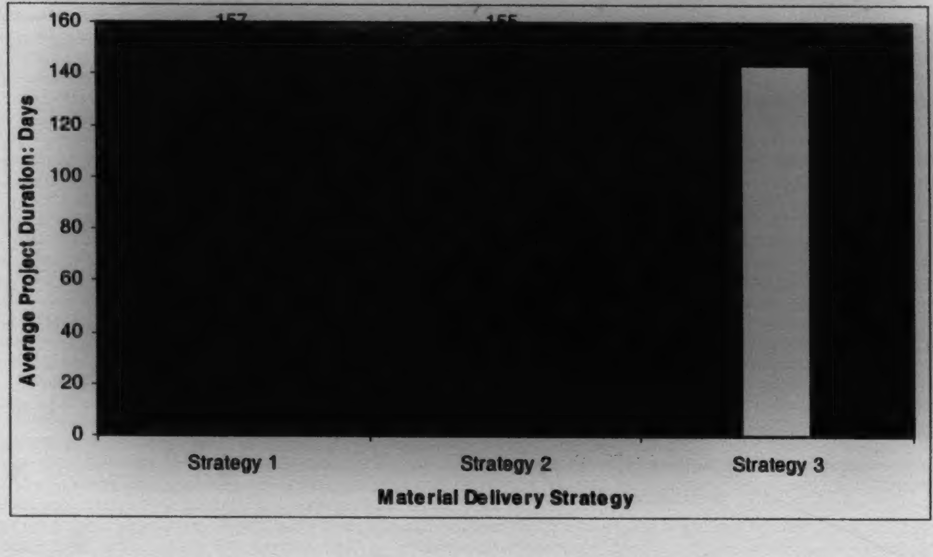


Figure 3-23. Analysis of Project Duration to Different Material Delivery Strategies

3.4.5 Increase Flexibility to Accommodate Downstream Uncertainty

Downstream uncertainties that arise during the execution of a project often cause frequent schedule changes and result in rework upstream. The uncertainties result from scope changes, site conditions, the unavailability of resources, and other unforeseeable events. Upstream flexibility, therefore, is required to accommodate downstream uncertainties. Real-time pull scheduling is a solution to improve upstream flexibility, which was discussed by Tommelein (1998). Tommelein's model, however, was simulated at a high abstract level without modeling the complex production system inside the shop and in the module yard, and could not therefore provide an updated schedule.

The created elements, *KanbanSender* and *KanbanReceiver* are used to embellish the developed model in order to model the real-time pull scheduling. The dynamic signals for requesting materials are sent from downstream to upstream to change the queuing priorities of drawings before they are drafted, or to change the queuing priorities of the spools before they are fabricated. The *TimeCollector* elements are set at various locations to trace the travel time of drawings and spools, and then to send collected information to the *DatabaseExporter* element. The information is then sent to the central database by the *DatabaseExporter* element in the end of simulation. Figure 3-24 illustrates the described logics. The dashed line indicates that the modeling elements are not physically connected in the model. The information is sent and received by identifying each other. The algorithm and codes can be referred to in the Appendix 2.

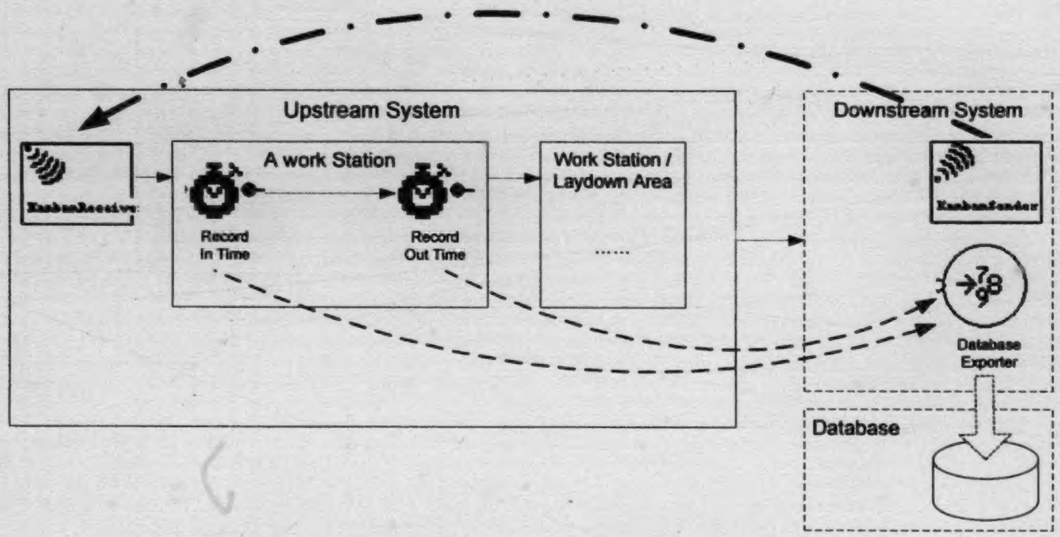
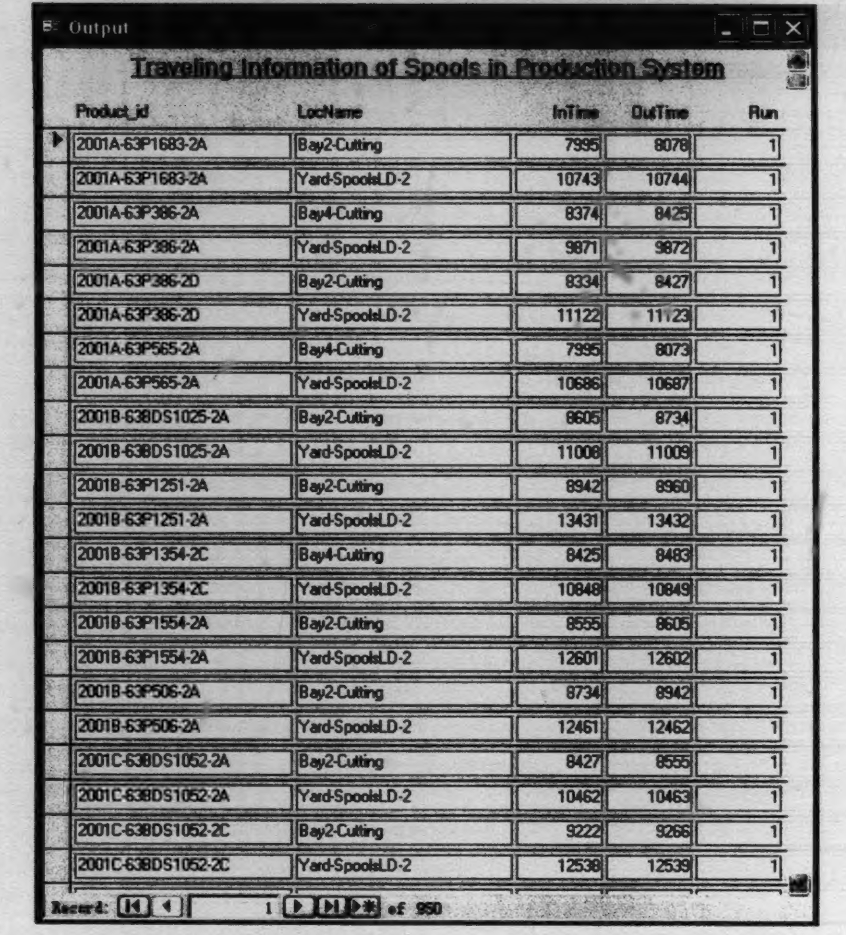

Upstream System
KanbanReceive
A work Station
Record
In Time
Record
Out Time
Work Station /
Laydown Area
……
Downstream System
KanbanSender
Database
Exporter
Database

Figure 3-24. Illustration of the Modeling Logics

Figure 3-25 shows the interface that retrieves the traced travel information for drawings and spools from the database.



Traveling Information of Spools in Production System

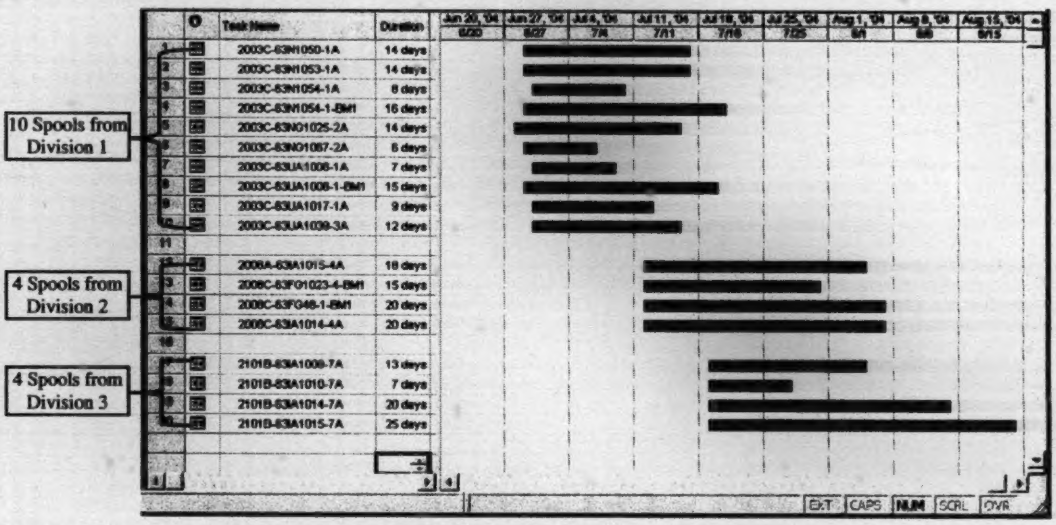
Product_id	LocName	InTime	OutTime	Run
2001A-63P1683-2A	Bay2-Cutting	7995	8078	1
2001A-63P1683-2A	Yard-SpoolsLD-2	10743	10744	1
2001A-63P386-2A	Bay4-Cutting	8374	8425	1
2001A-63P386-2A	Yard-SpoolsLD-2	9871	9872	1
2001A-63P386-2D	Bay2-Cutting	8334	8427	1
2001A-63P386-2D	Yard-SpoolsLD-2	11122	11123	1
2001A-63P565-2A	Bay4-Cutting	7995	8073	1
2001A-63P565-2A	Yard-SpoolsLD-2	10686	10687	1
2001B-638DS1025-2A	Bay2-Cutting	8605	8734	1
2001B-638DS1025-2A	Yard-SpoolsLD-2	11008	11009	1
2001B-63P1251-2A	Bay2-Cutting	8942	8960	1
2001B-63P1251-2A	Yard-SpoolsLD-2	13431	13432	1
2001B-63P1354-2C	Bay4-Cutting	8425	8483	1
2001B-63P1354-2C	Yard-SpoolsLD-2	10848	10849	1
2001B-63P1554-2A	Bay2-Cutting	8555	8605	1
2001B-63P1554-2A	Yard-SpoolsLD-2	12601	12602	1
2001B-63P506-2A	Bay2-Cutting	8734	8942	1
2001B-63P506-2A	Yard-SpoolsLD-2	12461	12462	1
2001C-638DS1052-2A	Bay2-Cutting	8427	8555	1
2001C-638DS1052-2A	Yard-SpoolsLD-2	10462	10463	1
2001C-638DS1052-2C	Bay2-Cutting	9222	9266	1
2001C-638DS1052-2C	Yard-SpoolsLD-2	12538	12539	1

Figure 3-25. Traced Travel Information of Spools

The information can be selectively queried and used to generate automatically a schedule in Microsoft Project.

In this case study, the original site installation plan is divided into three divisions. Due to uncertainties such as site availability, the construction sequence of these three divisions could change. The first experiment is conducted based on the original plan. The information detailing the time at which a spool enters the first work station, the cutting station, and leaves the last work station, the yard spools lay down area, can be queried from the simulation results and used to automatically generate a Microsoft Project file.

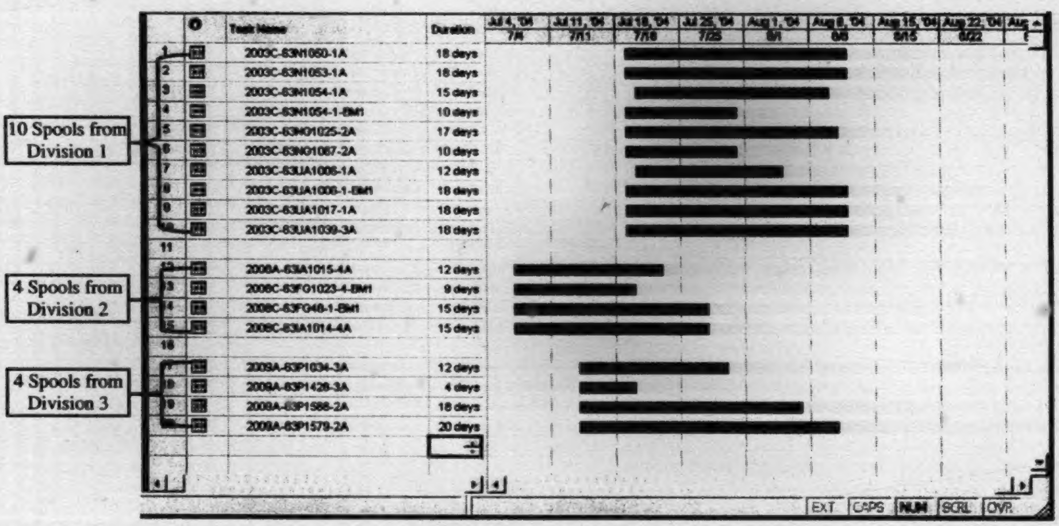
In order to illustrate the results clearly and to compare these results with the second experiment, 18 spools are chosen from the three divisions as the examples. This schedule is shown in Figure 3-26. The description in the task name is the spool number.


10 Spools from Division 1
4 Spools from Division 2
4 Spools from Division 3
Jun 20, '04
6/20
Jun 27, '04
6/27
Jul 4, '04
7/4
Jul 11, '04
7/11
Jul 18, '04
7/18
Jul 25, '04
7/25
Aug 1, '04
8/1
Aug 8, '04
8/8
Aug 15, '04
8/15
EXT
CAPS
NUM
SCRL
OVR

	Task Name	Duration
1	2003C-63N1050-1A	14 days
2	2003C-63N1053-1A	14 days
3	2003C-63N1054-1A	8 days
4	2003C-63N1054-1-BM1	16 days
5	2003C-63NG1025-2A	14 days
6	2003C-63NG1067-2A	6 days
7	2003C-63UA1006-1A	7 days
8	2003C-63UA1006-1-BM1	15 days
9	2003C-63UA1017-1A	9 days
10	2003C-63UA1039-3A	12 days
11		
12	2006A-63A1015-4A	18 days
13	2006C-63FG1023-4-BM1	15 days
14	2006C-63FG48-1-BM1	20 days
15	2006C-63A1014-4A	20 days
16		
17	2101B-63A1009-7A	13 days
18	2101B-63A1010-7A	7 days
19	2101B-63A1014-7A	20 days
20	2101B-63A1015-7A	25 days

Figure 3-26. Original Spool Fabrication Schedule

It is assumed that the changed site installation plan will change its sequence to be Division 2, Division 3, and Division 1. This change occurs a number of days after the project starts. The second experiment is conducted based on the changed plan. The simulation results are used to generate the updated schedule in Microsoft Project. In order to compare its results with those of the first experiment, the same 18 spools are selected for comparison; the results of updated spool fabrication schedule are shown in Figure 3-28.


10 Spools from Division 1
4 Spools from Division 2
4 Spools from Division 3
Jul 4, '04
Jul 11, '04
Jul 18, '04
Jul 25, '04
Aug 1, '04
Aug 8, '04
Aug 15, '04
Aug 22, '04
Aug
7/4
7/11
7/18
7/25
8/1
8/8
8/15
8/22
EXT
CAPS
NUM
SCRL
OVR

	Task Name	Duration
1	2003C-63N1050-1A	18 days
2	2003C-63N1053-1A	18 days
3	2003C-63N1054-1A	15 days
4	2003C-63N1054-1-BM1	10 days
5	2003C-63NO1025-2A	17 days
6	2003C-63NO1087-2A	10 days
7	2003C-63UA1006-1A	12 days
8	2003C-63UA1006-1-BM1	18 days
9	2003C-63UA1017-1A	18 days
10	2003C-63UA1039-3A	18 days
11		
12	2008A-63IA1015-4A	12 days
13	2008C-63FG1023-4-BM1	9 days
14	2008C-63FG48-1-BM1	15 days
15	2008C-63IA1014-4A	15 days
16		
17	2009A-63P1034-3A	12 days
18	2009A-63P1428-3A	4 days
19	2009A-63P1586-2A	18 days
20	2009A-63P1579-2A	20 days

Figure 3-27. Updated Spool Fabrication Schedule

The comparison shows that the schedule of spool fabrication changes due to the change of site installation plan. The spools of Division 2 and 3 are scheduled for fabrication before the spools of Division 1 in the updated schedule. The start and finish time are predicted in order to assist the production management staff in executing the daily operations. Another observation is that the durations of the same task in the two scenarios appear different. This is because the fabrication task is not a single activity. Many spools are processed by many resources at many workstations in the production system at the same time. The various queue delay times occurring at various locations can result in different fabrication times for the same spool in different scenarios.

In this case study, the simulation model can model the flexibility of the upstream systems in order to accommodate the uncertainty of downstream systems. When the changed plan of downstream system is known, the simulation model can be used to immediately generate the updated upstream production schedule.

3.4.6 Other Potential Experiments and Improvements

The developed simulation model is a model containing production-level details and which covers the entire project. It can be used to experiment with many ideas and strategies besides the above three examples. Other possible experiments include:

- ***Effect of labor efficiency***

Industrial construction is very labor intensive, involving many types of laborer. Laborers can be classified into different skill levels. The productivity of each activity can be estimated for the different labor skill levels. The model can be used to test the effect of labor efficiency upon the progress of a project.

- ***Effect of queuing rule***

Spool fabrication is a dynamic system. Spools are assigned to different work cells for fabrication based on certain criteria; however, a spool can be assigned to any of the work cells with same function only if the work cell is adequately capable. The queuing rule is not consistently followed, especially when the product family of spools is very unbalanced. In such cases, it often depends on the experience and ability of the foremen. Song (2004) considered this factor to be the supervision quality and compared this factor with four different rules: "random", "alternative", "shortest queue length", and "shortest waiting time". The model developed in this section can be used to test the effect of different queuing rules in a spool fabrication shop on the progress of a project.

- ***Yard layout optimization***

Spool fabrication facilities (for example, main fabrication shops, stress relief facilities, painting shops, or laydown areas) are scattered in the yard. These were built without optimization. Both transportation resources and time are involved in material/product movement. The model may be used to simulate different layout alternatives from which to choose the most efficient solution. The decision for a layout change, however, should be made together with a financial analysis of the reconstruction cost.

- ***Interaction of supply chains***

A large scale industrial construction project may have many subcontractors. A plant project being constructed in Edmonton, for example, has more than 30 contractors fabricating spools, steel structures, and modules. The interaction between these contractors is quite complex. The created modeling system can be used to model a dynamic supply chain system to test different project delivery strategies.

For all case studies, the simulation results are compared against those of experienced project managers and production engineers at the collaborating company. The developed simulation model made a reasonable analysis results.

3.5 CONCLUSIONS

This research designed and developed a special modeling system for building large scale simulation models for industrial construction projects. Construction engineers can use this modeling system to build simulation models efficiently for the whole supply chain system including drafting, material procurement and supply, spool fabrication, module assembly, and site installation with details at the production level. The modeling system addresses the characteristic of uniqueness in the product and information by modeling them in DBMS. The developed model is conceived in terms of not only activities, but also in terms of the product flow and information flow. Engineers can use the developed model to test various strategies in order to facilitate improving the system performance of industrial construction projects.

3.6 REFERENCES

AbouRizk, S. M., and Mohamed, Y. (2000). "Simphony - An Integrated Eenvironment for Construction Simulation", *Winter Simulation Conference Proceedings*, ed. Jeffrey A. Joines, Russel R. Barton, Keebom Kang, and Paul A. Fishwick, Orlando, Florida, pp.1907-1914.

Barrie, D. S., and Paulson, B. C. (1992). *Professional Construction Management, Including C.M., Design-Construct, and General Contracting*. McGraw-Hill, New York, NY, USA

Hajjar, D., and AbouRizk, S. (1999). "Simphony: an Environment for Building Special Purpose Construction Simulation Tools." *Winter Simulation Conference Proceedings,* ASCE, pp. 998-1006.

Hajjar, D., and AbouRizk, S. (2002). "Unified Modeling Methodology for Construction Simulation." *Journal of Construction Engineering and Management*, 128 (2) 174-185.

Halpin, D. W., and Kueckmann, M. (2002). "Lean Construction and Simulation." *Winter Simulation Conference Proceedings*, ed. Enver Yücesan, Chun-Hung Chen, Jane L. Snowdon, and John M. Charnes, SanDiego, California, Vol. 2, pp. 1697-1703.

Howell, G. A. and Ballard, H. G. (1996). *Managing Uncertainty in the Piping Process.* Research Report 47-13, Construction Industry Institute, University of Texas, Austin, TX, 103 pp.

Song, L. (2004). *Productivity Modeling for Steel Fabrication Projects*. PhD Thesis. Civil and Environmental Engineering Department, University of Alberta. Edmonton, Alberta, Canada.

Tommelein, I. D. (1998). "Pull-driven Scheduling for Pipe-Spool Installation: Simulation of Lean Construction Technique." *Journal of Construction Engineering and Management*, 124 (4) 279-288.

CHAPTER 4 – PHASE ONE RESEARCH CONTRIBUTIONS AND FINDINGS

4.1 INTRODUCTION

The work of Chapter 2 and Chapter 3 represents the first phase of the research. It involves many pioneering advances into the application of simulation technique and production theory in industrial construction. These contributions include: a pilot study for applying lean production techniques to spool fabrication; the development of a simulation-based approach to facilitate the implementation of lean techniques; the first simulation of a pipe spool fabrication shop, of its drafting process, of its information flow; the first model of an entire construction portion of project delivery of industrial construction on production level; the development of a special modeling system to model industrial construction projects; and a study addressing the product and information. These contributions are summarized in more detail in Chapter 7 together with Phase 2 research contributions.

Section 4.2 compares and discusses the performance of simulation models with different levels of detail and different scopes. Section 4.3 describes a virtual project management laboratory provided by the models developed in Phase 1. Phase 1 research greatly demonstrates the usefulness of production-based large scale construction simulation for improving the construction project system. The more important output of the Phase 1 research is the challenges identified in developing production-based large scale construction simulation. These challenges are discussed in detail in Section 4.4. Based on the discussion, conclusions are drawn in Section 4.5 to guide the Phase 2 research (Chapter 5 and Chapter 6).

4.2 DISCUSSION ON MODEL PERFORMANCE

- **Macro-Level Abstract Process Model**

Macro-level abstract process models are built and used by some researchers to simulate and study construction systems. These models are easily built, contain a small number of modeling elements, one layer structure, and consume less development time. Uncertainties at a lower level do not need to be studied and modeled in detail. Some researchers modeled the industry construction system, which is discussed in Chapter 3, using only about 15 to 20 modeling elements with one layer structure.

This type of model overly simplifies the internal complexity. For example, a sub-production system is often modeled as a single task in such a model. Resources, various products, and interactions are neglected, in spite of the fact that the most important task of a discrete simulation model of a production system is modeling and studying the interaction of activities and resources. Therefore, strictly speaking, a simulation model that does not model resources cannot be recognized as a simulation model. Furthermore, this type of model cannot depict accurately the performance of a system and reveal its problems.

The example of spool fabrication modeling can demonstrate the risk and unreliability of these types of model. In the simulation model for pipe-spool installation (Tommelein 1998), the fabrication system was modeled as a single activity with a duration of triangular distribution (3, 5, 14) and 20 fabrication crews; however, as described in detail in Chapter 2, a fabrication shop is a complex system and not a single activity. Many different types of resources work interactively on various tasks. The historical fabrication cycle time study (Figure 2-12 and Figure 2-13) indicates that the cycle time distribution cannot be represented as a simple triangular distribution. Many spools have a very long fabrication time. The total diameter inch of a spool is usually the most significant parameter for the productivity of fabrication activities such as cutting, fitting, and welding. Figure 4-1 is based on the same historical data used in the case study presented in Chapter 2. It does not, however, indicate correlation between the fabrication time of a spool and its total diameter inch. This issue becomes significant for the studied system, as every spool is unique. The delay of a certain spool might cause the delay in the progress of the whole assembly of a module or of the site installation.

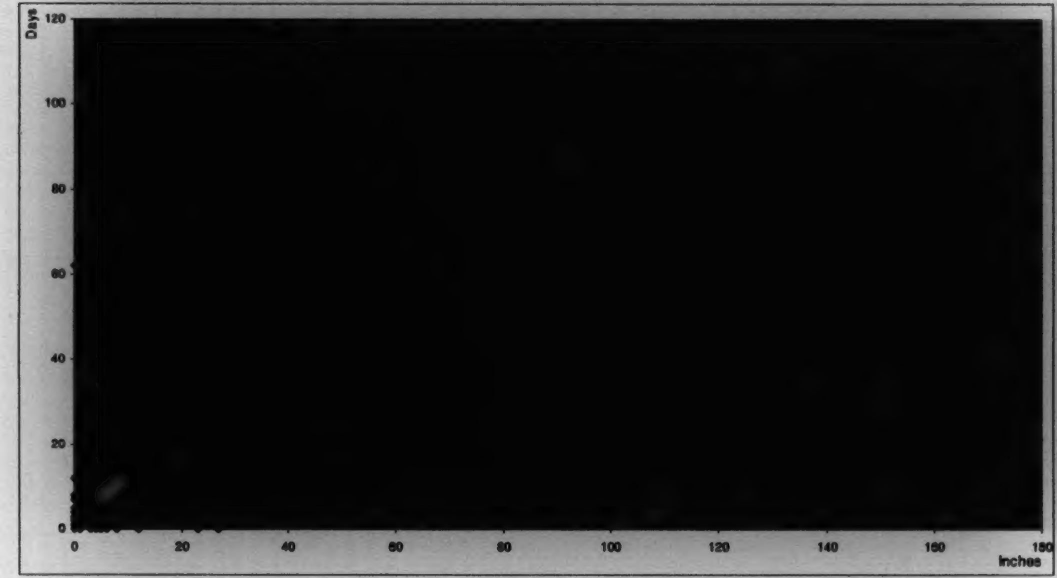


Figure 4-1. Relationship Between Fabrication Time and Total Diameter Inch of Spools

- **Isolated Component System Model**

Most construction simulation models have been limited to modeling the system of isolated components. These systems were improved or optimized for certain objectives such as reducing duration and increasing resource utilization. These achievements might not contribute to the objectives of the entire construction system due to the interference of other subsystems. In some cases, the intention to optimize a subsystem might not coincide with the objective of the entire construction system. For example, the objectives of a subsystem could be dynamic because of the requirements and uncertainties of other subsystems, and of the system as a whole.

The interaction between spool fabrication and module assembly illustrates well the above issue. Spool fabrication performance can be improved by modeling fabrication shop and experimenting scenarios, Module assembly can also be improved in this way. A problem, however, is between the two subsystems of spool fabrication and module assembly. Every spool is unique and will be installed in a certain position of a module; therefore, the delay of a spool through rework or by a change order may stop the assembly process of a module or the site installation. This problem often happens where the scheduling of site installation or of module assembly changes due to uncertainties. This will require an instant adjustment of the fabrication sequence for spools; thus a flexible schedule for spool fabrication should be an important objective to meet in accommodating the dynamic nature of downstream processes. An isolated component system model cannot offer an adequate solution.

- **Production-based and Project Scope Model**

A simulation model that simulates a construction system from the production level and covers the entire construction delivery cycle, behaves differently than the models described above. The models developed in Chapters 3 are of this type, and overcome the shortcomings of the former types of model. They can accurately capture and reflect product features, production processes, resource allocations and interactions, activity interactions, dependence, and variation. Only when a system is modeled from the production level, can construction engineers experiment with different scenarios and test all types of uncertainties at the production level. Various production-derived lean techniques can also be tested using such a model. A simulation model modeling the entire project delivery cycle allows construction engineers to experiment with different strategies of Lean Project Delivery System (LPDS) as well. Experiments and improvements that cannot be correctly executed in the above two types of model were achieved by the production-based large scale construction simulation model. Examples of these studies include the impact of drawing revision, effects of different material delivery strategies, and increases flexibility to accommodate downstream uncertainty. These experiments and improvements were depicted in Chapter 3.

4.3 A VIRTUAL PROJECT MANAGEMENT LABORATORY

Song (2005) discussed the essentials of the experimental planning approach and compared it with current production planning approaches in steel fabrication shops. Experimentation is the scientific approach used to search for cause and effect relationships in nature. The conclusions of these studies enable the researchers to apply findings to generate the desired effects. Song (2005) argued that the network-based production planning for a steel fabrication shop is more speculative than an experimental science. The research in industrial construction presented here proves Song's statement. Quantitative answers are extremely difficult to determine for certain issues in a complex and dynamic system with numerous uncertainties. Examples would include examining the risks and benefits of implementing lean production, the impact of drawing revision on the entire project delivery, and the impacts of uncertainties downstream on events upstream. Experiments in real construction projects, however, are highly risky and usually inefficient.

A production-based large scale construction simulation model can mimic the entire construction production system. It can then play the role of the virtual project management laboratory in which various planning experiments can be conducted. A project and its production system can be created and defined in an intuitive and graphical way. The model provides a laboratory allowing users to design, model, and evaluate any possible future scenario. Users perform "what-if" analyses and analyze "how-to-achieve" questions in the laboratory. Using a computerized environment rather than a real system makes systematic experimentation possible, easy, and affordable. It helps construction engineers to:

- Understand production performance of industrial construction;
- Test lean production/construction techniques, lean project delivery system strategies, and other theories with low cost;
- Test impacts of drawing revisions, material delay, rework, etc;
- Use simulation results to generate scheduling; and

- Use the simulation model as a marketing tool to help clients understand and trust the contractor's project delivery strategies.

4.4 IDENTIFIED CHALLENGES

The most important objective of phase one research is to identify the challenges in developing production-based large scale construction simulation and to find solutions to these challenges.

Simphony represents the start-of-the-art in construction simulation; many challenges were still faced, however, when building production-based large scale construction simulation models using this platform. Although phase one research made a number of contributions, the work in the phase one does not completely realize the vision depicted in Section 1.3. For example, the models do not utilize other worldviews aside from process interaction discrete event simulation. Also they are not open platforms that easily exchange information with other applications. This limited development is due in part to limited time and involvement invested by developers. The most substantial reasons, though, are the limited development efficiency and the limited versatility of current development platforms.

The following discussion is mainly based on the experience gained using the *Simphony* development platform. An identification of challenges was conducted through both development experience and theoretical analysis. Identified challenges include knowledge reuse, model decomposability, limited computing ability, product representation, information exchange with other applications, roots in activity scanning

(AS) or process interaction (PI) discrete event simulation, and one-world-view development. The following section discusses these challenges.

4.4.1 Lack of Domain Knowledge Capture, Standard, and Reuse

Developing a simulation model requires the collection of domain knowledge from construction domain experts. In order to develop the simulation models in Chapters 2 and 3, the author visited the cooperating contractor many times to interview the project managers, project coordinators, and superintendents; observe the production line, and study their information system. The author interviewed twelve industrial experts. An estimated 200 hours of accumulated time was directly spent on knowledge collection. This means that the model development will greatly depend on the domain experts, who are usually scarce and expensive. The knowledge acquisition is a time-consuming task. Nevertheless, there are more serious problems. The knowledge captured by the various developers is almost always different, as there is no standard currently used. Also the captured knowledge is not explicitly presented in the model. Moreover, the captured knowledge is not saved in a structured way and cannot be easily reused by future developers or modellers.

Developing a large scale model from scratch takes quite some time. The two research projects conducted in Chapters 2 and 3 lasted 1.5 years from the project initiation to the final completion. Tasks included system study, modeling system development, model development, and case study. (The author also worked on other research tasks during the period.) If the captured knowledge during development cannot be saved and reused by future developers in the same domain, it wastes effort. The lack of knowledge sharing also causes trouble for a development team in developing a large scale simulation project.

4.4.2 Lack of Model Decomposability

The construction system of an industrial construction project is large and complex. This type of project has multiple supply chains. A single model covering drafting, spool fabrication, module assembly, and site construction is too large to handle. The model in Chapter 3 is built by only one developer, the author. The author firstly needs to study each phase of drafting, fabrication, module assembly, and site construction to understand the production system. The author then works on the whole model throughout the development. The tasks of knowledge acquisition, model development, and data collection are all quite time-consuming. The time constraints limit the achievement of the phase one research. For example, the author does not have enough time to study the site construction and to develop the model as detailed in the fabrication shop model. There is also not enough time either for data collection or to model too many uncertainties.

Such a huge task should be achieved by a development team rather than a single developer. Furthermore, the model developed for a local process ought to be easily reusable in the future as a component to construct a new system model. This mainly depends on the development platform. The author conducted an investigation of the available tools. The construction simulation tools developed to date, however, lack good decomposability. The tools mentioned in Section 1.2.1 do not have decomposability functionality except *Simphony*. *Simphony* provides the decomposability to some extent through two features: modular hierarchical modeling structure and user elements. The modular hierarchical modeling structure allows a developer to construct a model layer by layer in a hierarchical manner. User elements are actually the libraries of developed components for reuse.

If a team had developed a production-based large scale construction simulation model using *Simphony*, the development task would have been executed as follows. The team decomposes such a model into several components. Each developer focuses on developing one of them without addressing other components. Each developer must have enough time to gain the knowledge of a subsystem and to develop an elegant model. Throughout the development, they need to agree with each other on the entities flowing between sub-models and on all attributes of the entities. They must keep each other informed if any changes are made.

Though the decomposability function has been achieved to a certain extent in *Simphony*, a few challenges or risks remain in this approach.

- The team does not have a specification for entities and their attributes. Likely, developers will have different perspectives on flowing entities in different subsystems. Moreover, required attributes will keep changing throughout the development. Though they can create a paper-based specification, it is not an integral object model of the system.
- Some modeling elements such as resources are valid globally. Using the same names for this type of modeling element in different component models will cause conflicts.
- The connection between component models is limited to the flowing entities. Other information cannot be shared among the component models.
- The size of the user element is very limited. The component models developed in Chapter 3, such as the fabrication plant model or the module assembly model, cannot be saved as user elements in *Simphony*, and then reused by future simulation developers.

- The developed large scale model cannot be distributed on multiple computers for execution.

4.4.3 Limited Computing Ability

Simulation is demanding on computing resources. Due to the uncertain nature of a construction project, a construction simulation model is typically run many times in order to acquire statistical results from all runs. Running a large scale model a number of times requires a lot of computing power. Additionally, more advanced graphic development and animation demands an even higher computing ability. In the case study explored in Chapter 3, one run of the model on a computer (Intel® Xeon™ CPU 3.06GHz; 1.00GB Rams; Microsoft Windows XP Professional Version 2002) took approximately 4 hours. If 30 runs were set up, it would take 120 hours. The project modeled in the case study is actually a small project consisting of only 687 spools. During the development, the model needed to be run many times for the purpose of testing. In this case study, it was run many times for different scenarios for the purpose of experimentation. Solutions to the challenge of computing ability should be developed, rather than simply waiting for advancements in computing hardware.

4.4.4 Lack of Product Presentation

Construction is not activity-driven, as discussed in Section 1.3. Construction produces products using certain resources. A facility to be constructed or any component of the facility to be fabricated can be viewed as a product. Current construction simulation models cannot contain abundant product information. Developers must use abstract entities to represent the product. Attributes can be added to entities in certain tools such as the common template in *Simphony*.

Product information, however, plays an important role in construction simulation modeling. Construction productivity is determined by many factors such as: material type, labourer skill, weather, and site conditions. Among them, the dominant factor is the physical properties of the product itself. The production process is also often determined by the product's physical properties. In a construction or fabrication system, products with different configurations often need to undergo different processes and require different resources. A typical example is pipe spool fabrication in industrial construction. Every spool is unique. A spool's fabrication process includes all or some of the processes of cutting, fitting, welding, stress relief, hydro testing, and painting. The processes are defined during the engineering or drafting phases. The production durations of these activities are different from one spool to another. When a model fails to capture product information, it fails to model a system successfully. The problem becomes more serious when the simulation model is used for planning purposes. When a real project is modeled to test and improve a project delivery system, it requires the product information of this particular project, instead of only historical information, since every project is unique.

In the modeling system of Chapter 3, the element *DatabaseImportor*, developed by the author using *Simphony*, can read the product and its information from the database to populate the simulation model. This information's accessibility improves the ability of the simulation tool to model construction products; however, the procedure by which the generated, exported, and manually collected product information is stored in a database is not systematic. The product information needed to be acquired keeps changing throughout development. The author experienced a difficult time determining the information and collecting the data. The determined production information for each type of product is defined using metadata in a database table. To date, none of construction simulation systems have a component providing a systematic means of modeling products.

4.4.5 Difficulties of Information Exchange with Other Applications

A production-based large scale simulation model must be driven by a large volume of information, including product information, resource information, cost information, job issue information, and site condition information. The information is usually available in other systems such as computer-generated drawings, scheduling systems, accounting systems, and Enterprise Resource Planning (ERP) systems. The simulation output can also be used by other applications, such as scheduling systems, if the model is powerful enough to realistically simulate a real project.

It is difficult to exchange data between the different systems sometimes because the simulation model file uses a different data format than the other applications. The model file is not human-readable either. To obtain information to build the models in Chapter 2 and Chapter 3 in order to execute the experiments, the author approached all of the above data sources. The simulation output is also utilized in the case study in Section 3.4.5. A lot of time and effort was spent on data acquisition and transformation; this also involved a high amount of manual work.

This issue can be extended to the whole construction industry, which is fragmented. It is widely known that it is difficult to exchange information between different applications due to the different data formats used by these applications in the construction industry. An open simulation platform and a human-readable neutral model or data format would be helpful for information exchange with other applications and will promote the application of simulation in the construction industry.

4.4.6 Roots in AS/PI Discrete Event Simulation

The modeling and simulation approaches completed to date have their roots in activity scanning (AS) or process interaction (PI) discrete-event simulations, which only represent one of the worldviews of simulation modeling. Thus the research successes achieved to date have been limited only to modeling the discrete-event construction processes. These models are incapable of answering questions beyond the scope of process interaction discrete-event simulation. Although all master production processes of a construction system can be simulated using AS or PI discrete-event simulation, some other components, such as complex uncertainties and their impacts, cannot be as easily modeled by AS or PI discrete-event simulation.

The uncertainties in a construction system fall into two categories: productivity uncertainty and process uncertainty. The high degree of uncertainty is the radical feature that distinguishes construction from manufacturing. Factors influencing productivity include site conditions, geologic conditions, weather, labourer skill, and various product configurations. For example, the productivity of module assembly and site installation is greatly influenced by weather, especially in very cold locations such as Alberta. Modeling weather conditions and its influence on productivity is significant. Process uncertainty refers to the immediate dispatching decision, changeable queuing rule, change orders, a client's imminent need, rework, etc. These factors interrupt the production system and make project scheduling vulnerable to change. For example, it can be very challenging to model a foreman's real-time decision to dispatch spools to work stations in order to achieve optimal shop performance in a fabrication shop. In order to model a construction system close to reality, however, these uncertainties cannot be neglected. The dynamic uncertainty variables and their influences on productivity and process often need to be modeled using advanced methods other than discrete-event simulation. For example, state-based modeling and continuous modeling could be used to model weather and site conditions, and constrained modeling, subjective modeling, and rule-based modeling could be used to model real-time decision-making.

It is necessary to expand the one-world-view of activity scanning or process interaction discrete-event simulation modeling to a "multi-world-view", which includes constrained modeling, subjective modeling, state-based modeling, and continuous modeling.

4.4.7 One View Development and Data Manipulation

Using current construction simulation tools, model developers can only manipulate the model from one view, graphically or code-wise. As the model grows bigger, problems emerge. For example, there is no central panel efficiently to control and manipulate data. During development of the model in Chapter 3, this problem became serious. It is a tedious and risky task to manipulate a large amount of data in such a large scale model. When a new scenario is needed for testing, the author must go into each element of the large modular hierarchical model in order to update the parameter value. When an error is made, it is often difficult to locate. In such cases a central information panel such as a resource pool is very helpful. It is necessary to explore multiple views to manipulate the model and the data in order to meet the various requirements for different cases.

4.5 CONCLUSIONS

In order to model, manage, and improve a construction system, the capacity of construction simulation models needs to be increased. In order efficiently to develop production-based large scale construction simulations, the cost-effectiveness of the development needs to be improved.

The above discussion concluded that the foundation for the objective of increasing the capacity of large scale construction simulation models and for improving the cost-effectiveness of its development are:

- Increasing knowledge standardization and reuse
- Model decomposability
- Computing ability
- Product representation
- Model openness
- Simulation world views
- Model development (or data manipulation) views

The relationship between the foundation and objective is shown in Figure 4-2. They are also the strategy used to guide phase two research in finding solutions for the identified challenges.

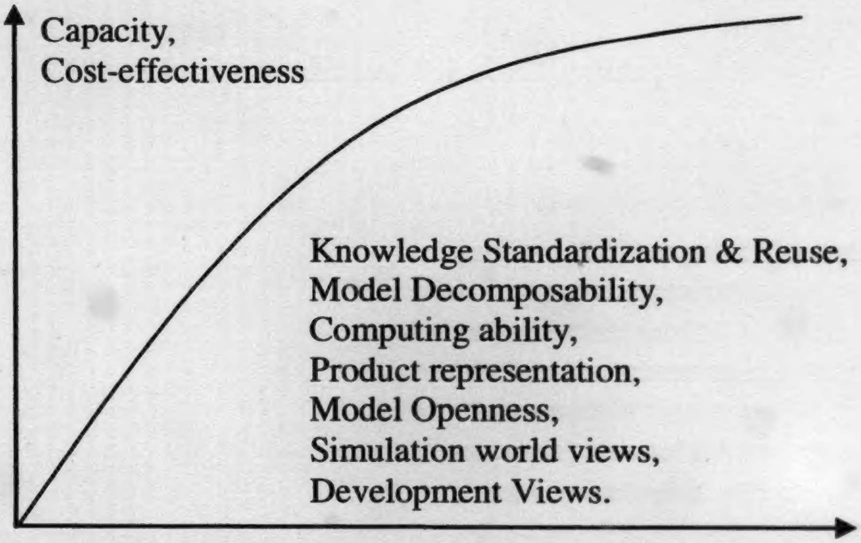


Figure 4-2. Foundation to Increase Capacity and Cost-effectiveness of Production-based Large Scale Construction Simulation Models

4.6 REFERENCES

Davis, D. D., and Holt, C. A. (1993). *Experimental Economics*. Princeton University Press, Princeton, NJ.

Song, L. (2004). *Productivity Modeling for Steel Fabrication Projects*. PhD Thesis. Civil and Environmental Engineering Department, University of Alberta. Edmonton, Alberta, Canada.

Tommelein, I. D. (1998). "Pull-driven Scheduling for Pipe-Spool Installation: Simulation of Lean Construction Technique." *Journal of Construction Engineering and Management*, ASCE, 124 (4), 279-288.

CHAPTER 5 – SOLUTIONS FOR IDENTIFIED CHALLENGES BY EXPLORING HLA, IFC, ONTOLOGY, AND XML

5.1 INTRODUCTION

In this chapter, the author attempts to find solutions that address the challenges simulationists face in production-based large scale modeling and simulation. A few relatively new theories and techniques such as High Level Architecture (HLA), Industry Foundation Classes (IFC), ontology, and eXtensible Markup Language (XML) contribute to the solutions. Sections 5.2 through 5.4 will introduce these theories, offering some background to the reader. The solutions to the identified challenges will then be explored in Section 5.5.

5.2 INTRODUCTION TO HLA

5.2.1 Origin of the HLA

In order to meet the simulation needs of defense-related projects, the Defense Modeling and Simulation Office (DMSO) of the Department of Defence of the United States developed HLA. HLA is now increasingly being used in other application areas such as transportation and manufacturing.

A complex simulation involves the combination of simulations for many different types of systems. Sometimes, the simulation of certain components has already been developed for other systems and can be used for the new simulation. In these cases, however, extensive modifications are normally necessary to enable the component simulation model being integrated into the new combined simulation. Often, it is easier to develop a completely new simulation of the system component, than to modify an existing one. This is because traditional simulation models lack two properties: reusability and interoperability (Kuhl et al. 1999). For example, a new missile system is to be installed into an existing aircraft. If simulation components of both the missile system and the aircraft have reusability and interoperability functions, the new missile simulation component can be integrated into an existing aircraft simulation to test a variety of simulation scenarios. It would save great time and effort not to have to develop a completely new simulation for the whole system.

The original purpose of HLA development was to meet the needs for standardizing the reusability and interoperability of developed simulation models. Reusability means that component simulation models can be reused in different simulation scenarios and applications. Interoperability means that the reusable component simulations can cooperate with other components without re-coding. Additionally, interoperability implies the ability to execute combined component simulations on a distributed network of computer platforms through a distributed, real-time operating system (Kuhl et al. 1999).

HLA is software architecture, not software. It is a specification for creating computer simulations using component simulations. It is a framework within which simulation developers can construct their simulation applications (Kuhl et.al. 1999).

A complex simulation can be considered as a hierarchical aggregation of components. The lowest level comprises the models of system components, which may be mathematical models, discrete-event queuing models, or rule-based models. The models are thus implemented in software to produce simulations. The simulation implemented as part of an HLA-compliant simulation is referred to as a federate. An HLA simulation is made up of a number of HLA federates and is called a federation (Kuhl et al. 1999).

5.2.2 Three Components of the HLA

1. HLA Rules

The HLA consists of a set of ten HLA rules at the highest level. They define standards to which federates and federations must adhere. Five of the rules pertain to federations and five pertain to federates.

The federation rules establish the ground rules for creating a federation (Kuhl et.al. 1999):

- *Rule 1 (Documentation requirements)*: Federations shall have an HLA Federation Object Model (FOM), documented in accordance with the Object Model Template (OMT).
- *Rule 2 (Object representation)*: In a federation, all simulation-associated object instance representations shall be in the federate, not in the Run-Time Infrastructure (RTI).
- *Rule 3 (Data interchange)*: During a federation execution, all exchange of FOM data among federates shall occur via the RTI.
- *Rule 4 (Interfacing requirements)*: During a federation execution, federates shall interact with the RTI in accordance with the HLA interface specifications.
- *Rule 5 (Attribute ownership)*: During a federation execution, an instance attribute shall be owned by at most one federate at a given time.

The federate rules deal with the individual federates (Kuhl et.al. 1999):

- *Rule 6 (Documentation requirements)*: Federates shall have a Simulation Object Model (SOM), documented in accordance with the HLA OMT.
- *Rule 7 (Control and transfer of relevant object attributes)*: Federates shall be able to update and/or reflect any attributes and send and/or receive interactions, as specified in their SOMs.
- *Rule 8 (Control and transfer of relevant object attributes)*: Federates shall be able to transfer and/or accept ownership of attributes dynamically during a federation execution, as specified in their SOMs.
- *Rule 9 (Control and transfer of relevant object attributes)*: Federates shall be able to vary the conditions (e.g. thresholds) under which they provide updates of attributes, as specified in their SOMs.
- *Rule 10 (Time management)*: Federates shall be able to manage local time in a way that will allow them to coordinate data exchange with other members of a federation.

2. *Interface Specification*

The interface specification defines a standard for a Run-Time Infrastructure (RTI). The RTI is software that conforms to this specification, and that provides common services to simulation systems. The RTI is the implementation of the HLA Interface Specification.

The RTI has 6 basic services (Kuhl et.al. 1999):

- Separates simulation and communication;
- Improves on older standards (e.g., DIS [Distributed Interactive Simulation], ALSP [Aggregate Level Simulation Protocol]);
- Facilitates construction and destruction of federations;

- Supports object declaration and management between federates;
- Assists with federation time management;
- Provides efficient communications to logical groups of federates;

The interface specification identifies how federates will interact with the federation and, ultimately, with one another. The interface specification is divided into 6 management areas:

- Federation management
- Declaration management
- Object management
- Ownership management
- Data Distribution management
- Time management

3. *Object Model Template (OMT)*

Reusability and interoperability require that all objects and interactions managed by a federate, and visible outside the federate, should be specified in detail and with a common format. The Object Model Template (OMT) provides a standard for documenting HLA object model information (Kuhl et.al. 1999). An outline of the OMT is shown below.

- Object Model Template (OMT)
  - Provides a common framework for HLA object model documentation.
  - Fosters interoperability and reuse of simulations and their components.
- Required Information
  - Object Class Structure Table

  - Object Interaction Table
  - Attribute/Parameter Table
  - Federation Object Model (FOM) / Simulation Object Model (SOM) Lexicon
- Optional Information (OMT Extensions)
  - Component Structure Table
  - Associations Table
  - Object Model Metadata

The OMT defines the Federation Object Model (FOM), the Simulation (or Federate) Object Model (SOM) and the Management Object Model (MOM). They are summarized below.

- Federation Object Model (FOM)
  - One per federation.
  - Introduces all shared information (e.g., objects, interactions).
  - Contemplates inter-federate issues (e.g., data encoding schemes).
- Simulation Object Model (SOM)
  - One per federate.
  - Describes salient characteristics of a federate.
  - Presents objects and interactions that can be used externally.
  - Focuses on the federate's internal operation.
- Management Object Model (MOM)
  - Universal definition.
  - Identifies objects and interactions used to manage a federation.

5.2.3 How the HLA Works

An HLA simulation is made up of a number of HLA federates and is called a federation. Federates, which are the component models, can be simulations, surrogates for live players, or tools for distributed simulation, as shown in Figure 5-1. A federate can model a number of entities. It can be a collector or a viewer of data. It can also be a surrogate for human participants in a simulation. Federates communicate through the RTI during execution (Kuhl et al. 1999).

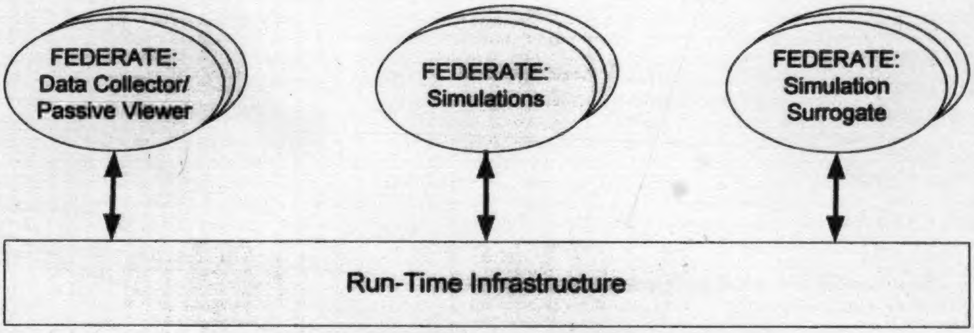

FEDERATE:
Data Collector/
Passive Viewer
FEDERATE:
Simulations
FEDERATE:
Simulation
Surrogate
Run-Time Infrastructure

Figure 5-1. Logical View of A HAL Federation

5.3 INTRODUCTION TO IFC

5.3.1 What is the IFC

Although many Vendor-specific IT applications in the Architecture, Engineering, Construction and Facilities Management (AEC/FM) industry have been developed (Staub et al. 1998), each of them is limited in coverage. An integrated solution is needed; however, it can only exist with standardized data models to support the exchange of information among disparate systems (Froese 1999).

The Industry Foundation Class (IFC) is a standardization effort, which was defined by the Industry Alliance for Interoperability (IAI) for sharing data throughout the project lifecycle, globally, and across disciplines and technical applications (IAI 1998). The IAI is a global, industry-based consortium for the AEC/FM industry (IAI 1996, IAI 1998). Its mission is to enable the computer applications used by all project participants to share and exchange project information. "The IFCs are used to assemble a project model in a neutral computer language that describes building project objects and represents information requirements common throughout all industry processes" (Froese 1999).

5.3.2 IFC Schema

The IFC has six major categories of concepts: *Project, Actor, Product, Process, Resource,* and *Relationships*. The first five concepts are fundamental to describing the major elements of construction knowledge.

IFC has four conceptual layers. The schema overview (Liebich 2004) of IFC2x Edition 2 is shown in Figure 5-2. Each layer defines a set of model schema. At any layer, an entity may reference another entity at the same or lower layer but may not reference a higher layer.

- ***Resource layer*** provides entities used by entities in the higher levels, eg. cost.
- ***Core layer*** provides the basic structure of the IFC object model and defines most general concepts that will be specialized by higher layers.
- ***Interoperability layer*** defines concepts (or classes) common to two or more domain models.
- ***Domain layer*** provides a set of modules tailored for specific AEC industry domain or application type.

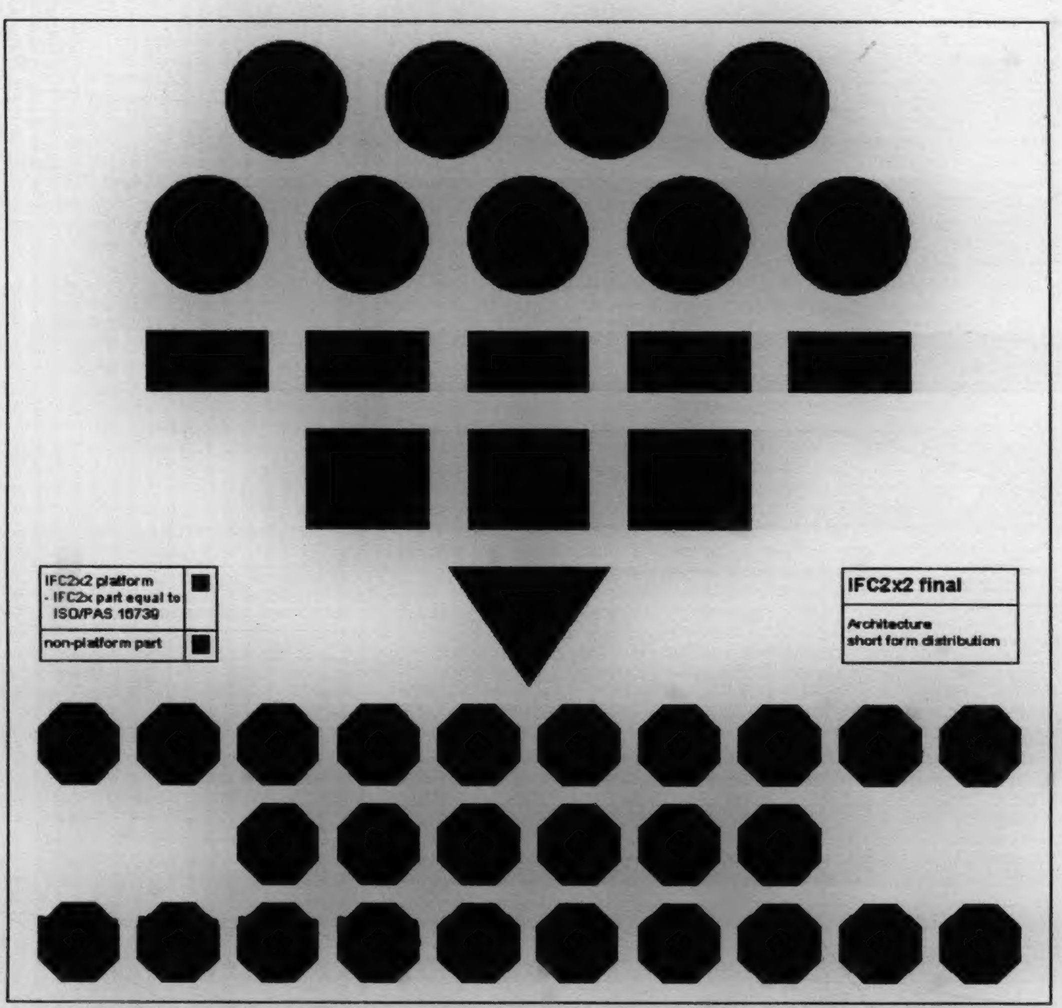

IFC2x2 platform
- IFC2x part equal to ISO/PAS 16739
■
non-platform part
■
IFC2x2 final
Architecture
short form distribution

Figure 5-2. Schema Overview of IFC2x Edition 2 (Liebich 2004)

5.3.3 ifcXML

ifcXML is the XML representation of the IAI IFC schema and data. The first schema of ifcXML was unveiled publicly in 2004.

XML has a broader range of supporting utilities and database implementations. It is the basis for most eCommerce messages and Web services. It is also supported by some web browsers. XML will be introduced in Section 5.5. "In offering an XML representation of IFC data, it is anticipated that a broader community of applications will be able to access a unified schema representing the built environment and related resources" (Nisbet and Liebich 2005).

ifcXML offers a combination of advantages. It uses XML technology for information publication and exchange and uses an internationally accepted and supported data standard (Nisbet and Liebich 2005).

5.4 INTRODUCTION TO ONTOLOGY

5.4.1 What is Ontology

The word *ontology* comes from the Greek *ontos*. In philosophy, it refers to the subject of existence. The purposes of creating ontology are sharing a common understanding of information among people, enabling the reuse of domain knowledge, and analyzing domain knowledge. It includes definitions of basic concepts in the domain and relations among them (Noy & McGuinness 2001). A key ability of ontology is to reduce semantic ambiguity for the purpose of sharing and reusing knowledge, and to create inter-operability.

Ontology was first used in the realm of Artificial-Intelligence. In recent years, the development of ontology has been moving to domain experts of many industries, such as World Wide Web, medicine, and wine. It is finding applicability in many areas of information system engineering such as database design and computer-supported collaborative working.

Ontology is machine-interpretable as well. So, it enables communication between computer systems in an independent way.

Ontology includes three main elements: taxonomies, associative relationships, and axioms. Taxonomies are sharable representations of vocabularies categorized in concept trees. They provide a shared terminology and define the meaning of each term in an unambiguous manner. Relationships link concepts across trees. Axioms support reasoning in a semantic way (El-Diraby et al. 2005).

5.4.2 Ontology Development for Construction Industry

The construction industry has a very low level of semantic system deployment (El-Diraby et al. 2005). The research of ontology development for the construction industry is fairly new.

Katranuschkov et al. (2003) developed an ontology framework as a gateway to simplify access to IFC model data. The structure of the framework is based on the XML schema specification. Staub-French et al. (2003) developed an ontology to support cost calculations. This ontology is an application example. It is dedicated to modeling the concepts and interrelationships in a specific domain. Turk (2002) analyzed the elements, which may be used to construct an ontology for construction IT.

El-Diraby is leading a project, e-COGNOS (COnsistent knowledGe maNagement across prOjects and between enterpriSes in the construction domain), to establish a formal ontology for the construction domain. The taxonomy has been developed to establish a skeleton of main concepts and a clustering mechanism. Its major root concepts are: *Project, Actor, Product, Process, Resource, and Technical Topics*. e-COGNOS was designed partially to reflect the IFC structure. The two systems have five common major domains (*Project, Actor, Product, Process, and Resource*). IFC terms were mapped to the e-COGNOS taxonomy (El-Diraby, et al. 2005). Based on the core concepts, sub-domain-level concept trees or application taxonomies can grow. A distributed ontology architecture was developed for highway construction (El-Diraby and Kashif 2005). Taxonomy for outside plant construction in a telecommunication infrastructure was developed (El-Diraby and Briceno 2005).

It is optimistically anticipated that ontology will be widely accepted in the construction industry and become the standard way for knowledge sharing and reuse as it has been in other industries. It will expedite the knowledge standardization in the construction industry. The ontology for the construction industry can be a source of standard construction knowledge used for construction simulation.

5.5 INTRODUCTION TO XML

5.5.1 XML and Its Emergence

Dick (2000) clearly analyzed the underlying reasons for the emergence of the XML format in his book. In short, there is a lack of understanding between data documents on the Internet. The traditional Internet technology is inadequate to address the following tasks: customized page layout, downloadable product comparisons, application integration, data integration, and interchangeable files. The root reason of the difficulty is that each party accessing a piece of information has its own perspective on the meaning of that information. The problem is particularly acute in some areas such as: web documents, electronic commerce, database access, and knowledge sharing. The information exchange problems of many areas converged to demand the finding of a general-purpose solution. Such a solution has to meet two requirements. One requirement is that a party can express the meaning of each piece of information. The other requirement is that two parties can spontaneously agree on the information organization (Dick 2000).

The general approach to meet the first requirement is adding metadata to documents. The field name of a database is an implementation. The approach to meet the second requirement is shared context, which are the rules metadata must follow. It serves as a contract among document users. XML is an implementation of the combination of the above general approach (Dick 2000).

XML was created by the World Wide Web Consortium (W3C) in 1998 as the descendent of Standard Generalized Markup Language (SGML), which is a standard of ISO. As the last generation of webpage language, Hyper Text Markup Language (HTML) is the descendent of SGML as well. However, HTML is only adequate for page layout, and does not address information exchange. The XML approach to implement metadata and shared context is to add metadata through tags and to add shared context through XML schema.

The whole purpose of XML is to exchange information. It is the standard for electronically exchanging data. XML emerged as a web technology for information exchange on the Internet. But it is much more than that, and used in many areas. Dick pointed out in 2000 that successful application of XML hinged on the support of different types of software, which are fundamental software components, software development tool support, document development tools, web infrastructure support, and translation components. In the last several years, this support was achieved. XML has gained increasing acceptance and is widely used.

The XML conceptual model has five components:

- Human and machine readability
- Defining content
- Defining structure
- Separation of content from relationships
- Separation of structure from presentation

XML is achieving wide application. For enterprises, it can be used for information distribution, knowledge management, workflow, application integration, and data integration. For software vendors, it can be used for customized publishing, information aggregation, software bill of materials, configuration and logging files, and distributed protocol (Dick 2000).

5.5.2 Benefit of XML to Simulation

XML is fairly new in the area of modeling and simulation, but it is increasingly being used. XML is used to develop neutral simulation models in the manufacturing industry (Mclean et al. 2002). The advantages and benefits of using XML to develop neutral models were discussed by Qiao (2003). The XML data model can be a neutral model used by different simulation packages. It provides an open platform for exchanging information with other applications. The National Institute of Standard and Technology (NIST) has developed a standard Shop Data Model using XML. A shop simulation model can be completely built using XML following the Shop Data Model. The neutral model can then be translated by a code generator into an executable simulation model in QUEST (a discrete event simulation tool developed by DELMIA). This simulation modeling methodology was implemented by Boeing (Lu et al. 2003).

5.[illegible] SOLUTIONS FOR THE IDENTIFIED CHALLENGES

5.6.1 Domain Knowledge Acquisition, Standardization and Reuse

Every construction project is unique. There are numerous uncertainties in construction projects, making construction simulation more difficult than manufacturing simulation. There is still, however, a large amount of common knowledge for all construction projects, especially for those in the same domain. The knowledge, which should be captured for reuse in construction simulation, includes product, information, resource, and process. The knowledge also includes to domain-independent construction environmental knowledge.

IFC and ontology are two standardization efforts; they can be the source of simulation-related construction knowledge and the knowledge structure, as depicted in Figure 5-3. The IFC is an internationally accepted standard for building projects in the industry of AEC-FM. Ontology is under research and development, and shows promise in playing a major role for knowledge standardization and knowledge sharing in the construction industry. Developed ontology for the construction industry will contain most knowledge required to develop a construction simulation model; this approach provides references for designing the simulation-related construction knowledge structure. It can also be the knowledge source for construction simulation.

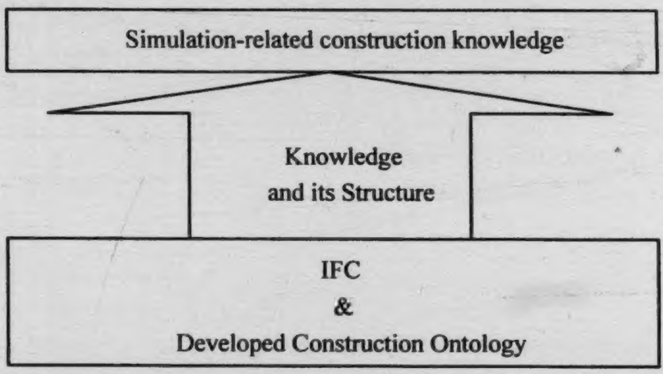

Simulation-related construction knowledge
Knowledge
and its Structure
IFC
&
Developed Construction Ontology

Figure 5-3. Acquire Standard Knowledge from IFC and Developed Construction Ontology

The construction simulation knowledge data will be fully object-oriented following the data structure of IFC and the developed ontology. After the captured standard knowledge is stored in the object model in a structured way, it can easily be reused by future developers and modellers. The multiple developers of a large scale model development team can also share the same knowledge without conflicts and confusion. The standard knowledge will be written in XML format, which will ease the information exchange with other applications.

5.6.2 Model Decomposability

The lack of model decomposability is actually due to the lack of interoperability and reusability of component models. HLA was developed to meet these two requirements. With interoperability and reusability, HLA makes simulation decomposable.

By using the HLA-compliant simulation tool, a development team can decompose a production-based large scale construction simulation model into several components of suitable size. The development task of a large scale model can thus be assigned to more than one developer. Each developer focuses on one subsystem without attending much to the other components. Finally, the developed models will cooperate with each other to form an HLA construction simulation model. Figure 5-4 depicts the procedure.

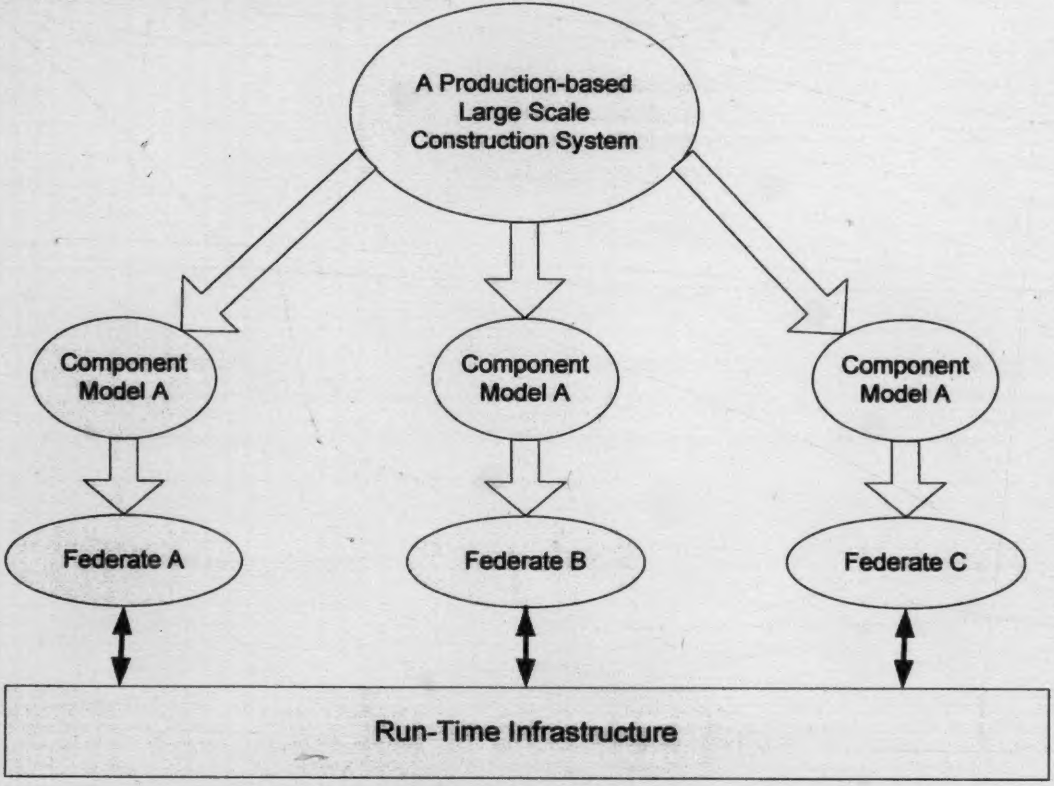

A Production-based Large Scale Construction System
Component Model A
Component Model A
Component Model A
Federate A
Federate B
Federate C
Run-Time Infrastructure

Figure 5-4. A Large Scale Model is Decomposed to Component Models

HLA not only makes it possible and easier to decompose and develop large scale simulation models, but also makes more efficient the re-use of the developed component models. Even though each construction project is unique, most subsystems of a construction system are repetitively used. Models of these subsystems developed as federates in HLA framework can be used in new models in the future.

A systematic way of decomposing a large scale construction simulation model has also been proposed. It can guide the decomposition and increase reusability. In conclusion, HLA results in higher efficiency and cost-effectiveness for production-based large scale construction simulation model development.

5.6.3 Computing Ability

Besides the advancement of computer hardware, there are several other solutions available to meet computing requirements. Examples include using a more advanced programming language to reprogram the simulation tool and optimizing the simulation algorithm.

Distributed simulation is another solution. This solution will be achieved when the HLA framework is successfully implemented. The HLA framework realizes the distributed simulation by allowing federates of a production-based large scale construction simulation model to be executed on multiple computers through a distributed, real-time operating system. This will greatly increase the simulation computing ability.

5.6.4 Product Model

Product information can be classified into three categories: work breakdown structure (WBS) information, physical property information, and production process information. These are generated during the engineering and drafting stages. Most product information is stored in engineering drawings, drafted shop drawings, and information systems (e.g. ERP) used by the construction industry.

An improved method using a database to store products information and the modeling element *DatabaseImportor* into import them to a simulation model was implemented in model development in Phase one research. However, there were still problems discovered.

First, the sources of product information have different data formats. In many cases, engineering drawings for one large project are produced by several companies using different CAD systems; even these CAD systems do not have shared data formats. Second, the product information contained in engineering drawings, drafted shop drawings, and information systems is much more than what is needed to develop a simulation model. At the same time, some of the needed information might still be unavailable from these systems. The relationship is demonstrated in Figure 5-5.

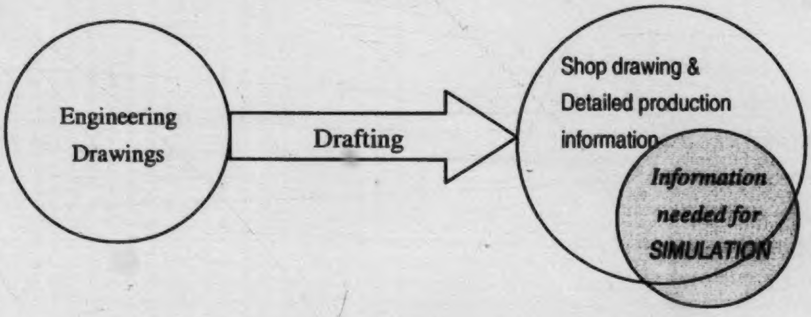

Engineering Drawings
Drafting
Shop drawing & Detailed production information
Information needed for SIMULATION

Figure 5-5. Product Information Generation

Therefore, it is not wise to export information from these systems to each simulation model. It would be more efficient if we could define and standardize what is needed for simulation modeling and present it in a structured way. If the structured presentation can gain wide recognition and can import information from any of these systems, it will ease future product modeling and make simulation modeling more efficient.

There exist several product models or construction concept models. Examples include CAD models, UniClass (UniClass, 1997), Talo90 ("Talo90" 1999), NAICS (NAICS

2003). None, however, particularly serve simulation purposes. Additionally, most of them are classification systems, instead of being object-oriented (El-Diraby et al., unpublished manuscript, 2005).

In Section 5.6.1, we propose to create a knowledge library for construction simulation. The product model will be a major component in the created knowledge structure and library. The fundamental product model structure is derived from IFC and the developed construction ontology. The product model creation is proposed for different types of projects since construction projects are very different from one type to another. The product model of a road product is completely different from an industrial plant product. Developing product models for each domain maintains the features and power of SPS (Special Purpose Simulation), which has been proven in *Simphony* (Hajjar and AbouRizk 2002). Product models will be created respectively for industrial construction, tunnelling projects, earth moving projects, road projects, and building projects. The created product models will be saved in XML format, which is neutral and human readable, to ease information exchange with other applications. Finally, extracting information from the computer-generated drawings or other applications to the standardized product model will be automated.

5.6.5 Information Exchange with Other Applications

The knowledge structure discussed in Section 5.6.1 will be presented by an XML schema, and the knowledge library will be stored in XML format. The construction simulation models will be saved as XML files as well. As pointed out in the section 5.5.1, the all purpose of XML format is to exchange information. Compared with a relational database, XML has three advantages: (1) ease of data exchange; (2) ease of transformation; (3) human readability. Utilizing this advantage of XML has taken place in manufacturing simulation.

The XML neutral data format of object models and simulation models eases information exchange for two purposes in construction simulation. One is the information exchange of construction simulation models with other applications in the construction industry. The other is the information exchange among different simulation packages. In the construction industry, however, the role of a neutral model for different simulation vendors is not important today because no construction simulation packages have been commercialized; it is not expected that many construction simulation vendors will emerge in the near future. Thus the focus is on the former purpose.

5.6.6 Multiple Simulation World Views

Research has been conducted to simulate weather conditions, geological conditions, productivity prediction, and decision making. These simulations follow different worldviews such as continuous, constrained, subjective, or rule-based modeling. The research effort now focuses on how to separate these models from master construction processes (discrete event simulation), make them public to combine with any master construction process, and reuse them easily.

The proposed solution is based on HLA–compliant simulation, which was introduced in Section 5.2 and discussed in Sections 5.6.2 and 5.6.3. The component model in the HLA framework is allowed to be any type of model: examples would include a mathematical model, a discrete-event queuing model, a rule-based model, or subjective model.

The author proposes these models ought to be initially separate from the master construction processes. Each of them can be developed as a federate or a few of them can be combined and developed into one federate. These federates will be saved in a library and will be plugged into a federation when needed. The federates can request information from any master process federate and their dynamic information can also be accessed by any other federate.

5.6.7 Multi-view Development / Data Manipulation

In the new simulation development environment, the model will be saved in XML format. An XML schema presenting structured construction simulation data may be built into the model. With the XML file of a simulation model, developers can view, develop, and manipulate the model from multiple views, rather than from only one view. Each view focuses on a subset of the model data to make the development or manipulation easier and more efficient. Different roles (advanced developers, developers, or users) can choose a different working view to develop, manipulate data, or to use the developed models, as illustrated in Figure 5-6. The possible views of model development or data manipulation include:

- Graphic Model Development View

This view is actually the same as the graphic development view of currently available construction simulation tools.

- XML Text View

This view is the XML text. All information of a simulation model is organized in a structured way in an XML text file. It is human-readable and editable. If a developer understands the XML schema and the model very well, he or she can directly develop the model or manipulate the data in this XML text view.

- Input/Output Views

Based on the XML model file, user-friendly interfaces can be developed for manipulating model input and output data. Possible applications include a central data control panel, layout configuration, cost view, and scheduling output.

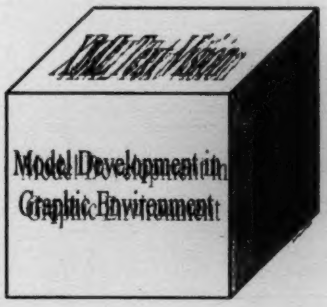


Figure 5-6. Multi-view Development / Data Manipulation of a Simulation Model

5.7 SUMMARY

Based on a wide volume of literature review and an investigation of IT advancement, its application in construction management, and simulation advancement in other industries (military and manufacturing), the author found several theories or techniques to solve the identified challenges. HLA, IFC, Ontology, and XML were introduced. Solutions to the identified challenges using these theories and techniques were discussed one by one from the theoretical perspective. To implement these theories and techniques together in one framework of production-based large scale construction simulation modeling will require more effort. This will be discussed in Chapter 6 as a prototype integrated architecture.

5.8 REFERENCES

Building 90 Group and the Finnish Building Centre Ltd. (1999). "Talo90: The Finnish building classification system." Helsinki, Finland.

Construction Project Information Committee. (1997). "UniClass-Unified Classification for the construction industry." RIBA Publications, UK.

Construction Standards Insititute. (2003). "Overall Construction Classification System (OCCS)." < www.occsnet.org > (May 10, 2005).

Dick, K. (2000). *XML: A Manager's Guide*. Addison-Wesley Longman Inc., Massachusetts, USA.

El-Diraby, T. E., and Briceno, F. (2005). "Taxonomy for Outside Plant Construction in Telecommunication Infrastructure: Supporting Knowledge-Based Virtual Teaming." *Journal of Infrastructure Systems*, Vol. 11, No. 2, pp. 110-121.

El-Diraby, T. E., and Kashif, K. F. (2005). "Distributed Ontology Architecture for Knowledge Management in Highway Construction." *Journal of Construction Engineering and Management*, Vol. 131, No. 5, pp. 591-603.

El-Diraby, T. A., Lina, C., and Feis, B. (2005) "Domain Taxonomy for Construction Concepts: Toward a Formal Ontology for Construction Knowledge." *Journal of Computing in Civil Engineering*, Volume. 19, Issue. 4, pp. 394-406

Fishwick, Paul A. (1995). *Simulation Model Design and Execution – Building Digital Worlds*. Prentice Hall, Englewood Cliffs, New Jersey, USA.

Froese, T., Fischer, M., Grobler, F., Ritzenthaler, J., Yu, K., Sutherland, S., Staub, S., Akinci, B., Akbas, R., Koo, B., Barron, A., and Kunz, J. "Industry Foundation Classes

For Project Management-A Trial Implementation", *Electronic Journal of Information Technology in Construction*, Vol.4, 1999, pp.17-36.

Hajjar, D., and AbouRizk, S. (2002). "Unified Modeling Methodology for Construction Simulation." *Journal of Construction Engineering and Management*, 128 (2) 174-185.

IAI (1996). "End User Guide to Industry Foundation Classes, Enabling Interoperability in the AEC/FM Industry", International Alliance for Interoperability (IAI).

IAI (1998). "International Alliance for Interoperability (Home Page) [online]". <http://www.interoperability.com>. Accessed March 22, 1999.

Katranuschkov, P., Gehre, A., and Scherer, R. J. (2003). "An Ontology Framework to Access IFC Model Data." *Electronic Journal of Information Technology in Construction*, Vol.8, p. 413-437.

Kuhl, F., Weatherly, R., and Dahman, J. (1999). *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*. Englewood Cliffs, NJ: Prentice Hall.

Liebich, T. (2004). "IFC2x Edition 2 Model Implementation Guide." International Alliance for Interoperability Modeling Support Group.

Lu, R., Qiao, G., and McLean, C. (2003). "NIST XML Simulation Interface Specification at Boeing: A Case Study." *Proceedings of the 2003 Winter Simulation Conference*, ed. Stephen E. Chick, Paul J. Sánchez, David Ferrin, and Douglas J. Morrice, 1230–1237.

Mayer, R. J., Menzel, C. P., Painter, M. K., deWitte, P. S., Blinn, T., and Perakath, B. (1995). *IDEF3 Process Description Capture Method Report*. <http://www.idef.com/idef3.html> (Nov. 15, 2004).

McLean, C., Jones, A., Lee, T., and Riddick, F. (2002). "An Architecture for a Generic Data-Driven Machine Shop Simulator." *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes. 1108–1116.

NAICS (2003). "North America Industry Classification System Association (NAICS) (Home Page) [online]" <http://www.naics.com>. Accessed June 10, 2005

Nisbet, N., and Liebich, T. (2005). "ifcXML Implementation Guide." International Alliance for Interoperability Modeling Support Group.

Noy, N. F., & McGuinness, D. L. (2001). "Ontology Development 101: A Guide to Creating Your First Ontology." <http://www.ksl.stanford.edu> (Dec. 5, 2004)

Qiao, G., Riddick, F., and McLean, C. (2003). "Data Driven Design And Simulation System Based on XML." *Proceedings of the 2003 Winter Simulation Conference*, ed. Stephen E. Chick, Paul J. Sánchez, David Ferrin, and Douglas J. Morrice, 1143–1148.

Staub-French, S., Fischer, M., Kunz, and Paulson. (2003) "An ontology for relating features with activities to calculate costs." *Journal of Computing in Civil Engineering*. ASCE. Vol. 17 (4).

Turk, Ž. (2002). "Elements of an Ontology of Construction Informatics." *Proceedings of the European Conference on Information and Communication technology Advances and Inovation in the Knowledge Society - eSMART 2002*, Rezgui Y, Ingirige B, Aouad G (ed.), University of Salford, 155-167.

Walsh, N. (1998). "A Technical Introduction to XML." <http://www.xml.com> (Dec. 10, 2004).

CHAPTER 6 – PROTOTYPE INTEGRATED ARCHITECTURE

6.1 INTRODUCTION

The solutions for identified challenges were discussed in Chapter 5. Several theories and new techniques were explored to solve the challenges. The reader may have noticed overlaps among the proposed solutions. This is because the author found and proposed solutions by targeting the challenges separately. Each of the identified challenges may require more than one methodology or technique to be resolved. Simultaneously, each of the methodologies or techniques may contribute to solving more than one problem. Table 6-1 demonstrates their relationships.

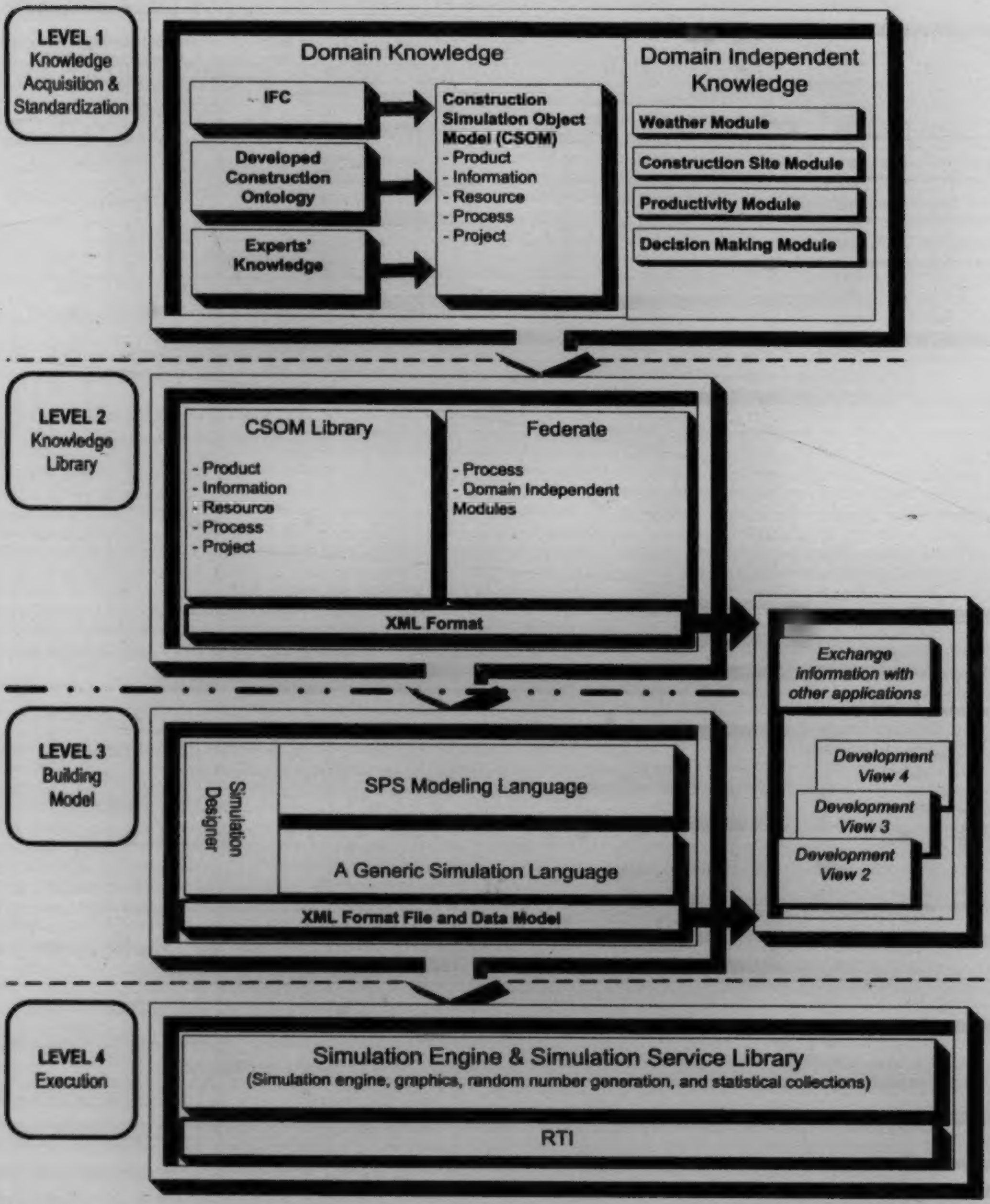
Table 6-1. Relationships between Challenges and Methodologies/Techniques

	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
HLA		✓		✓		✓	
IFC	✓		✓		✓		
Ontology	✓		✓		✓		
XML	✓		✓		✓		✓

Instead of implementing the solutions separately, the proposed solutions are integrated into one framework. Integrating these solutions poses a more challenging job because of the issue of the compatibility of these techniques. For example, the HLA has certain specifications for the object model. The object model exported from IFC or from the developed construction ontology must be transformed to meet the requirements of HLA specifications.

A prototype integrated architecture for developing large scale construction simulation models is proposed by the author and shown in Figure 6-1. This proposed architecture has four levels and a side level. Sections 6.2 through 6.6 describe these levels. This research focuses mainly on Level 1 and Level 2 because these two levels are more related to construction industry knowledge. Considerable effort is spent on the side level as well since similar line of knowledge has not been explored by other researchers in the construction industry. The concepts behind Level 3 and Level 4 are also described; however, their developments are beyond the scope of this thesis and are pursued by other researchers working with the NSERC/Alberta Construction Industry Research Chair simulation team at the University of Alberta. The outcome of their development is expected to meet the needs of the proposed architecture.

Although the full development of the platform is beyond the scope of the thesis, part of the system has been accomplished by the author. A case study to implement the proposed architecture is conducted based on the available components of the proposed system and illustrated in Section 6.7. Conclusions are drawn in Section 6.8.



LEVEL 1
Knowledge Acquisition & Standardization
Domain Knowledge
IFC
Developed Construction Ontology
Experts' Knowledge
Construction Simulation Object Model (CSOM)
- Product
- Information
- Resource
- Process
- Project
Domain Independent Knowledge
Weather Module
Construction Site Module
Productivity Module
Decision Making Module
LEVEL 2
Knowledge Library
CSOM Library
- Product
- Information
- Resource
- Process
- Project
Federate
- Process
- Domain Independent Modules
XML Format
Exchange information with other applications
Development View 4
Development View 3
Development View 2
LEVEL 3
Building Model
Simulation Designer
SPS Modeling Language
A Generic Simulation Language
XML Format File and Data Model
LEVEL 4
Execution
Simulation Engine & Simulation Service Library
(Simulation engine, graphics, random number generation, and statistical collections)
RTI

Figure 6-1. Architecture of Production-based Large Scale Construction Simulation Modeling

6.2 LEVEL 1: KNOWLEDGE STRUCTURE AND ACQUISITION

This level collects knowledge about domains of interest in a structured way. This is the main feature of the proposed architecture that differs from current construction simulation practices.

6.2.1 Knowledge to Acquire

First, we need to know what knowledge is required to develop a production-based large scale construction simulation model. Construction projects can be classified into different types such as earth moving projects, tunnelling projects, and high-rise projects. No matter what type of project it is, the knowledge needed for the construction simulation includes: product, information, resource, and process. We call this domain-dependant knowledge. There are other types of knowledge, such as weather or site condition, which are not domain related. We call this domain-independent knowledge. The knowledge needed for construction simulation is identified and classified as follows:

- **Domain-Dependent Knowledge**

➢ ***Product***

As discussed in Section 4.4.4 and Section 5.6.4, product information plays an important role in production-based construction simulation modeling, but is not well modeled or standardized in current practice. Product knowledge becomes the major component of domain-dependent knowledge in the proposed architecture.

There are many types of products. The variance depends on the project type: it could be a wall or a story in a building project, or a spool in industrial construction.

Simulation-related product knowledge can be classified into three categories:

- Work Break Down (WBS) information

  E.g. job number, control number, and component ID in a facility.

- Physical property information

  E.g. weight, diameter, and material type.

- Production process information

  E.g. needs cutting, needs fitting, and does not need painting.

➢ ***Information***

Information flow was not modeled in most of past construction simulation; however, it is an important entity flowing through the construction production system. Information processing is a special type of production process. It also influences other production processes such as interrupting a production system. For example, the process of drafting shop drawing is the upstream of material procurement and fabrication. It directly determines the start of the downstream process and the quality of fabrication. These should not be neglected in a production-based construction simulation. Information flow was modeled for the first time in the Phase one of this thesis, but not in a standard way. Information knowledge becomes another component in domain-dependent knowledge.

There are two types of information in construction simulation:

1. Drawing

   E.g. ISO drawing and shop drawing;

2. Communicating Document

   E.g. Non-conformance Report (NCR), Request For Information (RFI), and the Kanban in lean construction theory.

The important simulation-related knowledge about information is the WBS and priority.

- ***Resource***

Resource is the basic knowledge needed for construction simulation. Resource can be anything needed for constructing a project. Three major categories are:

1. Labourer
2. Equipment
3. Space

Resource knowledge is not modeled in detail in current construction simulation practice. Resources are usually modeled through limited states, mostly two states, the “idle” state and the “busy” state. In the proposed architecture, more knowledge can be added to the resource. A resource can carry more attributes, such as capacity. It can have multiple states besides “idle” and “busy”.

- ***Process***

Process knowledge is the central knowledge for construction simulation. Processes are normally considered and modeled during the model building stage based on the knowledge collected from industry.

Different domains have different processes; however, every domain has a set of basic processes and each process has a set of basic activities. The IFC model and the developed construction ontology (e.g. e-COGNOS) contain process knowledge. Some other organizations, such as CII, have developed standard construction process knowledge as well.

Simulation-related process knowledge includes: process ID, process name, duration, required resources, cost, WBS information, logic with other processes, and relationships with products and information.

- **Domain-Independent Knowledge**

Domain independent components are separate from domain-dependent master knowledge. The domain-independent components include: weather models, construction site condition models (e.g. ground condition or soil type), productivity prediction models, etc. Models of domain-independent components can be continuous, constrained, or artificial intelligent models. It is quite difficult to find a generic methodology to describe all of them. They need to be analyzed and modeled separately. Fortunately, they have a high reusability and can be reused and shared by all master processes of any domain.

Domain-independent knowledge is not the focus of this thesis research.

6.2.2 Knowledge Structure

The knowledge structure was designed following three principles:

1. Object-oriented;
2. Compatible with internationally accepted industrial standards;
3. Maintain the features of SPS.

The knowledge structure is based on HLA-compliant simulation requirements, an investigation of the IFC data model, and the developed construction ontology. It is named the Construction Simulation Object Model (CSOM). CSOM is a hierarchical object-oriented model. There are two layers in the knowledge structure.

Core layer:

The core layer defines the objects, which are common to all types of construction projects. It includes:

- Product
- Information
- Resource

Each object has some attributes, which are common to all types of projects. For example, a "resource" object of any type of project has the attribute *cost*. Figure 6-2 shows the three basic object classes. All object classes will inherit from these three object classes. The knowledge of process is not included in the CSOM because the process knowledge is complicated and processes have relationships with other types of objects. The process knowledge will be modeled and stored as part of the federates' development. The objectification of process knowledge and automatic translation into a simulation model is further discussed in Section 7.3.3.

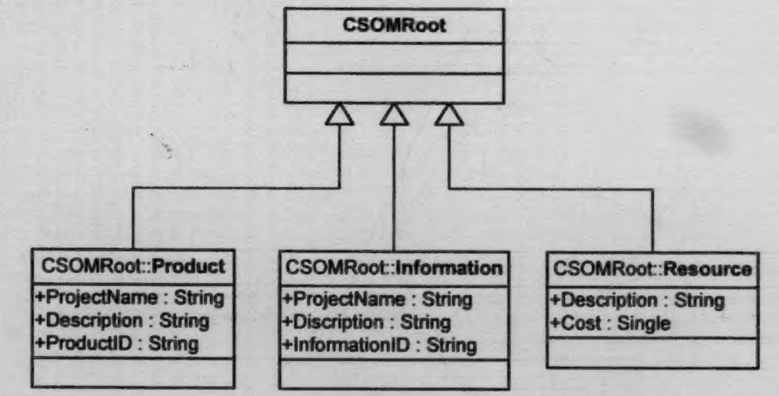

CSOMRoot
CSOMRoot::Product
+ProjectName : String
+Description : String
+ProductID : String
CSOMRoot::Information
+ProjectName : String
+Discription : String
+InformationID : String
CSOMRoot::Resource
+Description : String
+Cost : Single

Figure 6-2. Core Layer of CSOM

Domain-specific layer:

The knowledge is acquired and structured for each type of construction project. Construction projects are classified into the following domains:

- Earth moving
- Tunnelling
- Road
- Mining
- Building
- Industrial construction

Knowledge structure is developed for each domain. All objects of the domain of interest inherit from the three object classes of the core layer. The domain layer contains specific objects taken from the domain of interest. For example, in industrial construction, spool, steel piece, and module are specific kinds of products. The domain layer may have more than one hierarchical level. The attributes of these domain-specific objects are defined as well. Figure 6-3 is an example of part of the Unified Modeling Language (UML) static model of industrial construction object classes.

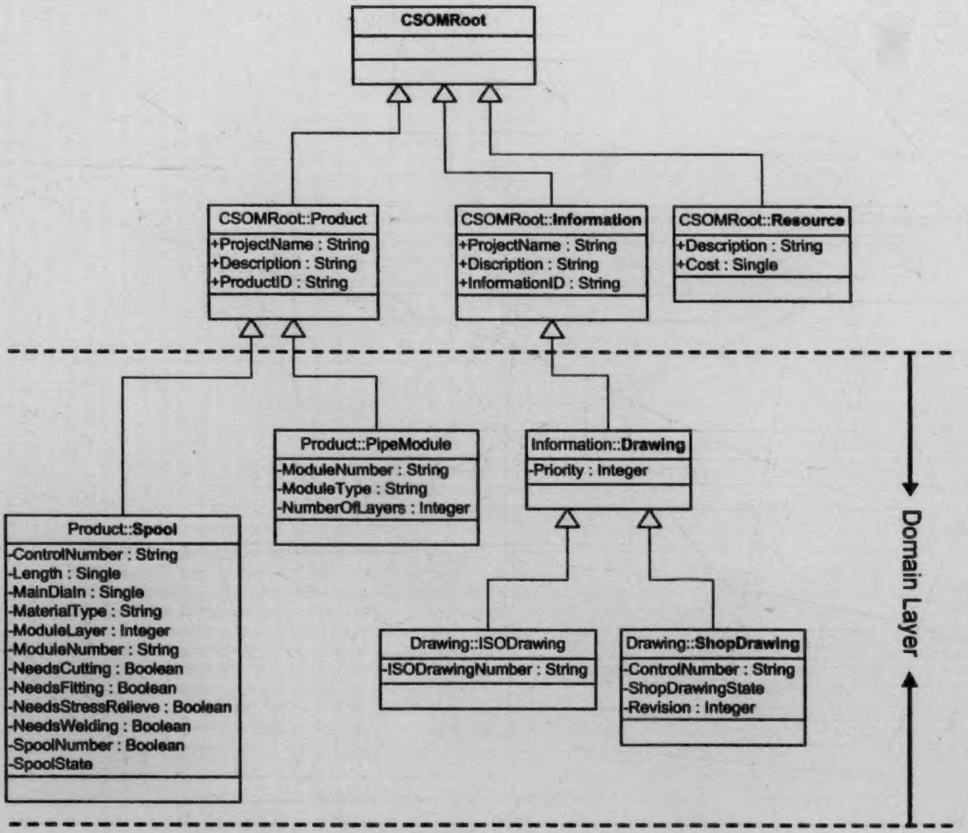

CSOMRoot
CSOMRoot::Product
+ProjectName : String
+Description : String
+ProductID : String
CSOMRoot::Information
+ProjectName : String
+Discription : String
+InformationID : String
CSOMRoot::Resource
+Description : String
+Cost : Single
Product::PipeModule
-ModuleNumber : String
-ModuleType : String
-NumberOfLayers : Integer
Information::Drawing
-Priority : Integer
Product::Spool
-ControlNumber : String
-Length : Single
-MainDiaIn : Single
-MaterialType : String
-ModuleLayer : Integer
-ModuleNumber : String
-NeedsCutting : Boolean
-NeedsFitting : Boolean
-NeedsStressRelieve : Boolean
-NeedsWelding : Boolean
-SpoolNumber : Boolean
-SpoolState
Drawing::ISODrawing
-ISODrawingNumber : String
Drawing::ShopDrawing
-ControlNumber : String
-ShopDrawingState
-Revision : Integer
Domain Layer

Figure 6-3. Domain-specific Layer of CSOM

6.2.3 Three Ways of Knowledge Acquisition

There is not a single available source that can provide all standard knowledge needed for construction simulation. We need to approach several sources to obtain the standardized knowledge to build the knowledge library for production-based large scale construction simulation for all domains of interest.

- **Import object model from ifcXML**

The most recently released IFC was written in XML, and is referred to as ifcXML. The proposed CSOM library is written in XML as well, and will be discussed in detail in

Section 6.3.1. This provides the opportunity to transform the data model from ifcXML to CSOM.

The IFC model was developed mainly for building projects for the AEC-FM industry. Therefore, we import object models from the IFC to the domain of the building project in CSOM.

The IFC model scope covers the data needed for the entire life cycle of the building project, and for all disciplines involved over the life cycle. However, it is clear that no single application will ever need to support all data for all disciplines over the complete life cycle. The subsets of the IFC model that comply with specific exchange data needs at particular stages of a building life cycle phase are defined as "views" (Nisbet and Liebich 2005). The data needed for construction simulation is only a small subset of the IFC model; in other words, it is only one view on the IFC model.

eXtensible Stylesheet Language Transformations (XSLT) is a language for transforming XML documents into other XML documents. An XSLT can be written to generate an "IFC model view" for construction simulation from ifcXML as illustrated in Figure 6-4.

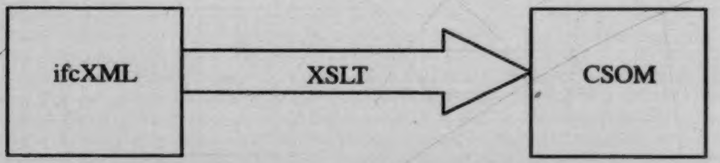


Figure 6-4. Import Objects Model from ifcXML to CSOM

- **Import object model from developed construction ontology**

While IFC was developed especially for building projects, ontology is being developed to cover different types of construction projects. The developed ontology includes highway construction (El-Diraby and Briceno 2005) and outside plant construction in a telecommunication infrastructure (El-Diraby and Kashif 2005). They are written in WOL (Web Ontology Language). The developed ontology is fully object-oriented. The object model can also be exported to CSOM. The data needed for construction simulation is a subset of the ontology model. An XSLT can be written to transform the developed construction ontology into CSOM as illustrated in Figure 6-5.

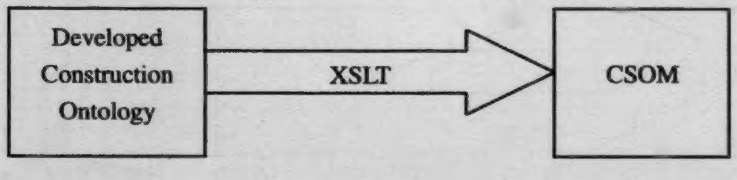


Figure 6-5. Import Object Model from the Developed Construction Ontology to CSOM

For example, based on the available developed ontology, we can import an object model from highway construction to the domain of the road project in CSOM. As discussed in Section 5.4, however, the research of ontology in the construction industry is in its initial stage. It does not provide many knowledge sources for construction simulation; however it is predicted to play the most important role in knowledge standardization in the construction industry in the near future. More and more domain-specific ontology will be developed. The developed construction ontology will be an important knowledge source for construction simulation.

- **Model experts' knowledge**

The available internationally accepted standards, such as IFC and developed ontology, are good sources to generate object models for construction simulation; however, they do not cover all types of construction projects. When we cannot find such standard knowledge, we need to collect the knowledge from experts in the construction industry, and to model the knowledge and document it in a formal way.

Unified Modeling Language (UML) notation can be a tool to document the acquired knowledge. The information of the generated UML model can be easily transformed by XML Metadata Interchange (XMI) Export into the XML format CSOM model, as illustrated in Figure 6-6.

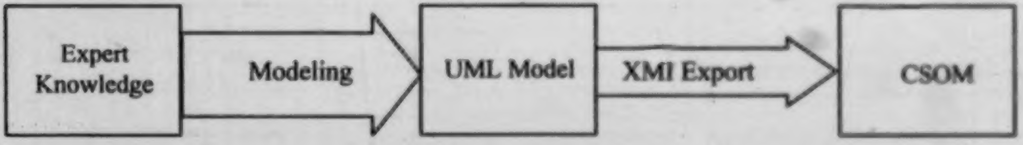


Figure 6-6. Model Experts' Knowledge Using UML to Generate CSOM

Another means is modeling the acquired knowledge directly in an XML editor, as illustrated in Figure 6-7. The editor also acts as the knowledge library repository. It is introduced in detail in Section 6.3. For example, the author built simulation models for spool fabrication and the whole industrial construction system in Chapters 2 and 3. Since an internationally accepted standard object model for industrial construction was not found, the knowledge acquired from industrial experts can be modeled using an XML editor.

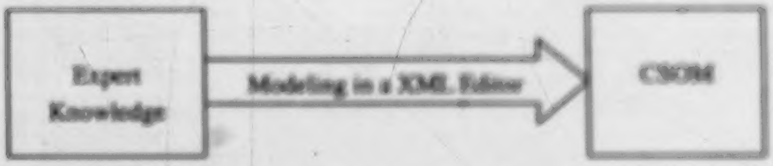


Figure 6-7. Model Experts' Knowledge Using XML Editor to Generate CSOM

6.2.4 Knowledge Tailoring

The knowledge acquired from the IFC or a developed ontology may not be exactly the knowledge that simulation models need. Knowledge that is unrelated to simulation can be pruned from the library; likewise, knowledge needed for simulation can be added to the library.

The knowledge acquired from industry experts needs tailoring as well because the standard cannot be created by only one developer at one time; rather, it is an iterative procedure that needs modification and maintenance along with development practice.

6.3 LEVEL 2: STRUCTURED KNOWLEDGE LIBRARY

The modeling activities of Level 1 must have the support of Level 2. The acquired knowledge needs to be stored as structured objects or federates in a library in order to be used to develop simulation models.

6.3.1 Extensible Object Model Library

An object model editor is needed to edit and store the CSOM. There are a few commercial XML editors available on the market such as XMLEditPro. In this research, the editor developed by the NSERC/Alberta Construction Industry Research Chair simulation team for the HLA object model is chosen to accommodate the CSOM. The advantage of using this tool instead of other XML editors is that it meets the compatibility requirement of the HLA object model. Figure 6.8 shows the editor with an object library. The hierarchical structure on the left demonstrates the structure of the CSOM. The right side defines the object in detail. The objects are stored in XML format.

This basic structure is recommended as the knowledge object standard for production-based large scale construction simulation. The content is extensible for future developers. More knowledge can be accumulated and stored in this library as construction simulation continues to develop.

The following is an example of the object models acquired from domain experts. The author acquired knowledge of the industrial construction from industry experts, and developed object models for industrial construction. Figure 6.9 is the sample object model viewed with the object model editor.

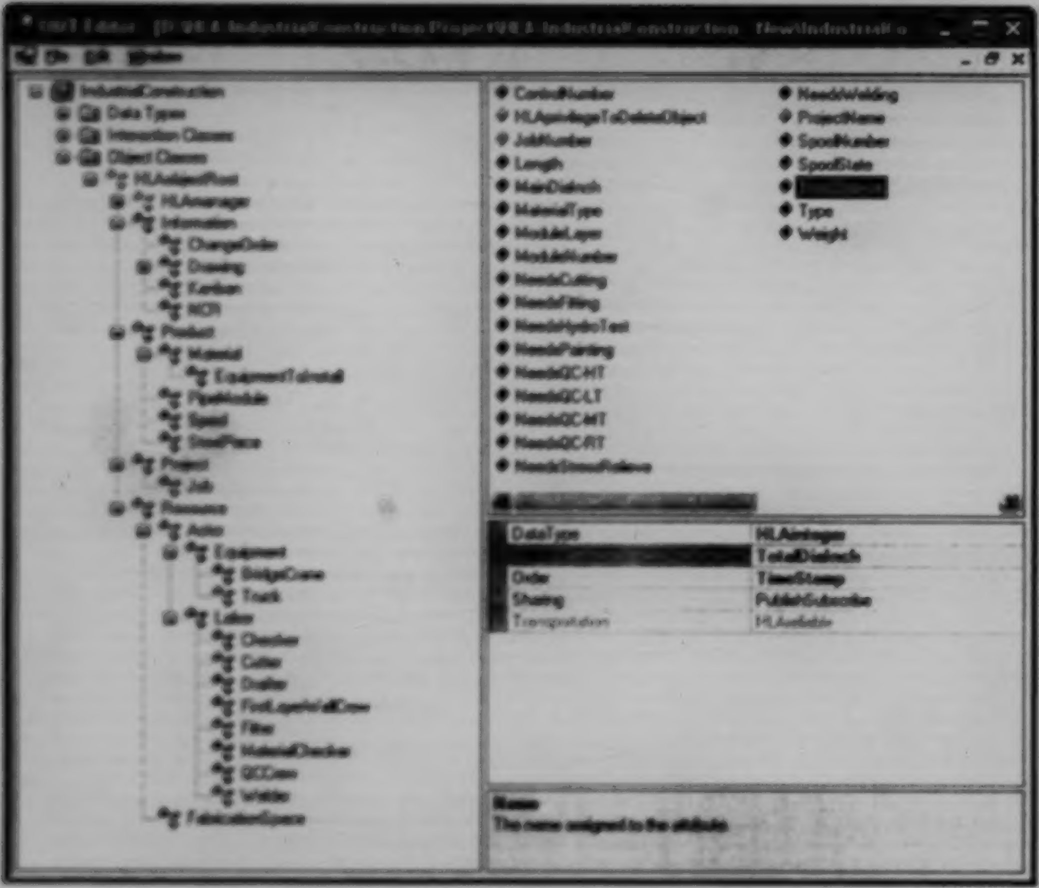


Figure 6-8. Sample CSOM for the Domain of Industrial Construction

6.3.2 Process Model Library

This is achieved together with Level 3. One of the most important purposes of HLA-compliant simulation is its reusability. The typical master processes of each domain can be developed as a basic structure of federates for the domain and stored as a process library. The developed federates can be saved for reuse in new projects in the future. This methodology makes the construction process modeling effort more feasible.

6.3.3 Domain-Independent Model Library

The models for domain-independent components will be developed and saved as federates within the HLA framework. HLA federates allow the component model to be any type of simulation. The model libraries will be accumulated along with the simulation development practices. The developed federates can be reused to cooperate with all domain-dependant master process models. Any federate can be dragged and plugged into a new HLA simulation model to build a new federation together with domain simulation models. They have high reusability.

Nevertheless, the development of domain-independent component models is not the focus of this thesis research. The models and libraries will be developed by other researchers involved in HLA construction simulation research.

6.3.4 Advantages of Knowledge Standardization and Library

- **Compatible with internationally accepted standard**

The knowledge structure is derived from internationally accepted standards. It is compatible with the data model structure of IFC and with the existing ontology of the construction industry. It enables construction knowledge data to be imported from the existing IFC and ontology models.

- **Reusability**

The acquired construction simulation-related knowledge is going to be standardized. The library exists independently from simulation models, but is easily accessible by the simulation models. This greatly increases the reusability of the knowledge.

- **XML format**

The library is saved in XML format. It eases data exchange with other applications. The library permits generating FOM and SOM through XSLT.

- **Object-oriented**

In contrast to classification systems, the knowledge library is an object-oriented model; it allows for future expansion. The object-oriented approach also provides for greater flexibility in modifying concepts independently.

- **Extensibility and modifiability**

The library is extensible; that is, the knowledge library can be extended along with simulation development practices. The knowledge structure is designed based on the author's extensive investigation and research, and is recommended as the standard construction simulation-related knowledge structure for building production-based large scale construction simulations. However, better ideas and designs arising from future research may improve the structure; therefore, the library model allows for such modifications.

6.4 LEVEL 3: BUILDING A MODEL

The model building level of the architecture contains a simulation building designer. This function employs a simulation language consisting of a set of modeling elements. It is capable of modeling construction systems. When a developer builds a simulation model using the simulation language, object instances can be created from the saved knowledge object library at Level 2 and be used to construct a model at Level 3.

6.4.1 Modeling Elements

Two basic types of modeling elements are needed at Level 3 to construct a simulation model. One type is for discrete event simulation. All discrete event simulation tools, such as Visual Slam, CYCLONE, QUEST, and Simpony, provide this type of modeling elements. The other is for dealing with communication among component models. They exist especially in HLA-compliant simulation tools.

A set of modeling elements are developed for discrete-event simulation as part of Simphony.HLA by the NSERC/Alberta Construction Industry Research Chair simulation team. Examples include ResourceElement, TriggerQueueElement, CaptureElement, TraceElement, SetOutputElement, ReleaseElement, and HoldElement (NSERC/Alberta Construction Industry Research Chair 2005). Another set of modeling elements are developed to deal with communications among federates by the same team. Examples include DiscoverElement, ObserverElement, AcquireElement, UpdateElement, and DivestElement (NSERC/Alberta Construction Industry Research Chair 2005). In this research, these developments are chosen to build discrete-event simulation models and to deal with communications among component models because they were developed with the need for compatibility with the HLA framework.

6.4.2 Generate Specific eFOM

An FOM defined by HLA specifications only contains the objects shared among federates. The FOM in this architecture is extended to include all objects of product, information, and resource regardless of whether they are shared among federates or not. It is defined as an extended FOM (eFOM). An XSLT can be written to generate a specific eFOM for developing a simulation federation from the comprehensive knowledge library, CSOM, at Level 2, as illustrated in Figure 6-9. The generated eFOM is then imported to the simulation model as reference.

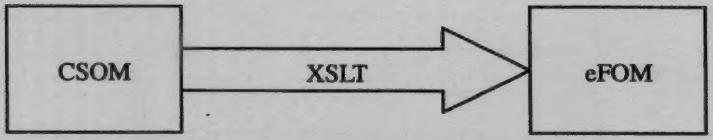


Figure 6-9. Generate eFOM from CSOM

The generated eFOM serves as the knowledge object library to provide all standard object classes needed for building a simulation model. Object instances can be created from the object class of the eFOM and used to construct the simulation model.

6.4.3 HLA-compliant Simulation Model Development

An example of HLA-compliant simulation model development can be found in the proposed redevelopment of the model of an industrial construction project described in Chapter 3. We decompose it into its component models of drafting, spool fabrication, module assembly, and site construction. The development task is then assigned to four developers. Each component model can be developed as a federate by one developer.

The modeling elements are used to build each of these federates. The object instances of product, information, and resources are created from the object class of eFOM. The master processes of the industrial construction domain are imported into these federates as well. The domain-independent component models can also join the federation if needed. Finally a HLA-compliant simulation model called a federation is developed.

Owing to the reusability and interoperability of HLA-compliant simulation, these developed federates will make up the model library. These model libraries are used to develop new projects, and are illustrated in Figure 6-10.

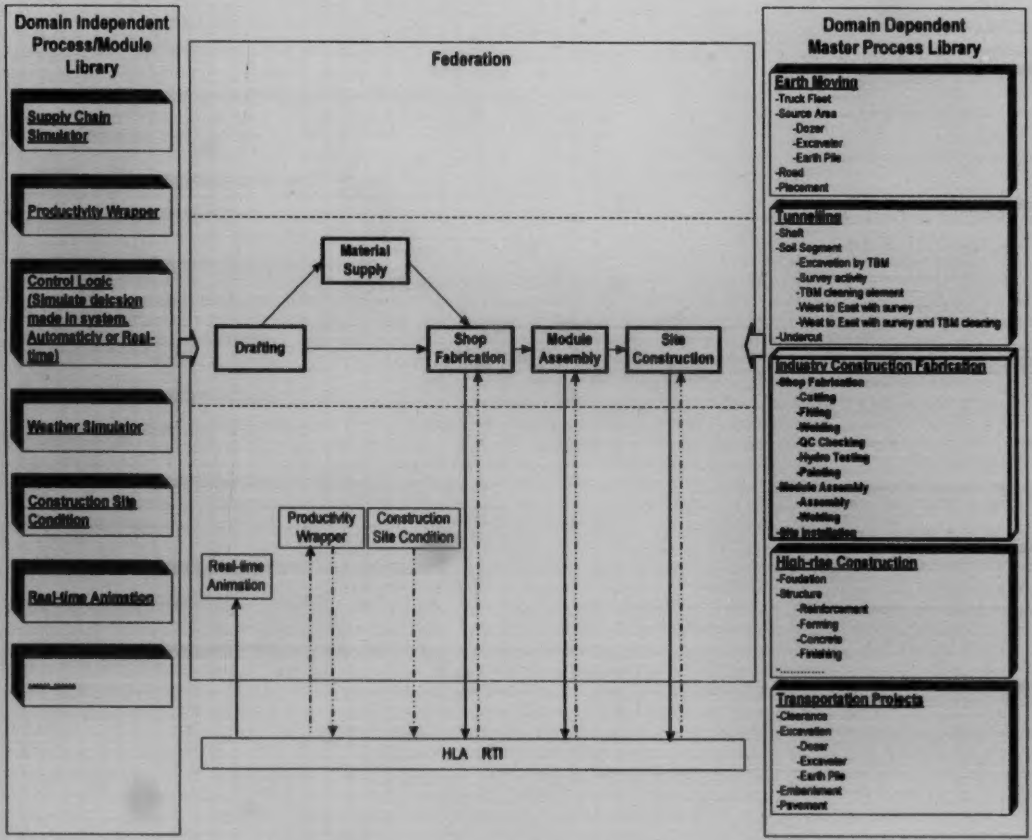

Domain Independent Process/Module Library
Supply Chain Simulator
Productivity Wrapper
Control Logic (Simulate deicsion made in system, Automaticly or Real-time)
Weather Simulator
Construction Site Condition
Real-time Animation
Federation
Material Supply
Drafting
Shop Fabrication
Module Assembly
Site Construction
Productivity Wrapper
Construction Site Condition
Real-time Animation
HLA RTI
Domain Dependent Master Process Library
Earth Moving
-Truck Fleet
-Source Area
-Dozer
-Excavator
-Earth Pile
-Road
-Placement
Tunnelling
-Shaft
-Soil Segment
-Excavation by TBM
-Survey activity
-TBM cleaning element
-West to East with survey
-West to East with survey and TBM clearing
-Undercut
Industry Construction Fabrication
-Shop Fabrication
-Cutting
-Fitting
-Welding
-QC Checking
-Hydro Testing
-Painting
-Module Assembly
-Assembly
-Welding
-Site Installation
High-rise Construction
-Foudation
-Structure
-Reinforcement
-Forming
-Concrete
-Finishing
Transportation Projects
-Clearance
-Excavation
-Dozer
-Excavator
-Earth Pile
-Embankment
-Pavement

Figure 6-10. Build Simulation Model Using Model Library

6.5 LEVEL 4: EXECUTION

The execution level refers to low-level simulation services, which include basic simulation service and HLA simulation service. A simulation model built at Level 3 is executed using the services provided in Level 4. A production-based large scale construction simulation model can be executed on multiple computers.

6.5.1 Simulation Services

The basic simulation service includes a calendar manager, simulation engine, random number generation, tracing, and statistical collection and analysis. These basic simulation services are available in any discrete-event simulation tool such as VisualSlam, CYCLONE, QUEST, and *Simphony*. These services are being recoded by the NSERC/Alberta Construction Industry Research Chair simulation team to fit into the new HLA-compliant simulation platform, Simphony.HLA. The developed services are chosen for this research.

Another basic simulation service is the RTI for HLA-compliant simulation. The RTI provides the software services necessary to support an HLA-compliant simulation. During execution, the component models communicate through an RTI using standard HLA-compliant protocols. IEEE (The Institute of Electrical and Electronics Engineers) 1516-2000 standard is the most current standard specification used to develop the RTI. Different versions of the RTI are possible. One version is available from Defense Modeling and Simulation Office (DMSO) and can be downloaded free of charge from the DMSO website. The NSERC/Alberta Construction Industry Research Chair simulation team developed its own version of RTI, *Simphony* RTI, based on the IEEE 1516-2000 standard (NSERC/Alberta Construction Industry Research Chair 2005), and is the version used in this research.

6.5.2 Execution on Multiple Computers

The models built at Level 3 are finally executed through the services provided at Level 4. The component simulation models, federates, can be executed on different computers with communication through RTI, which is located on server as illustrated by Figure 6-11.

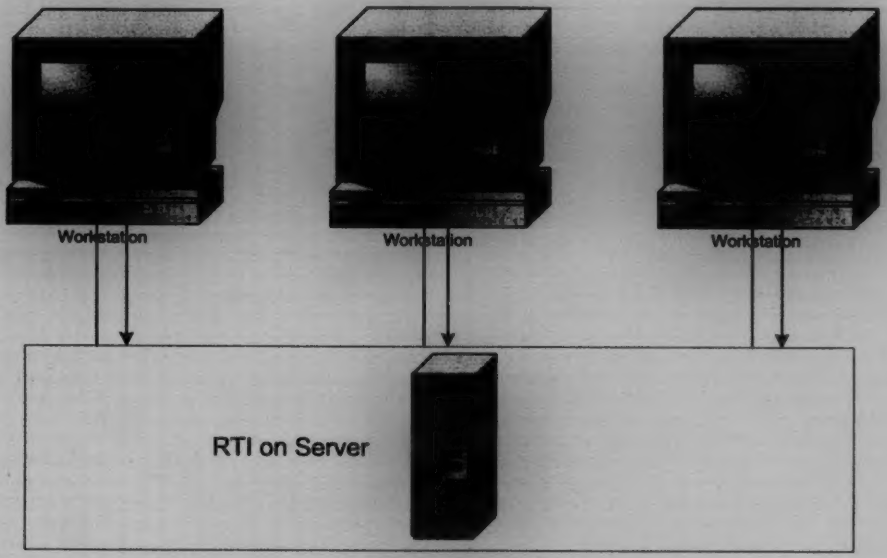


Figure 6-11. Execution on Multiple Computers

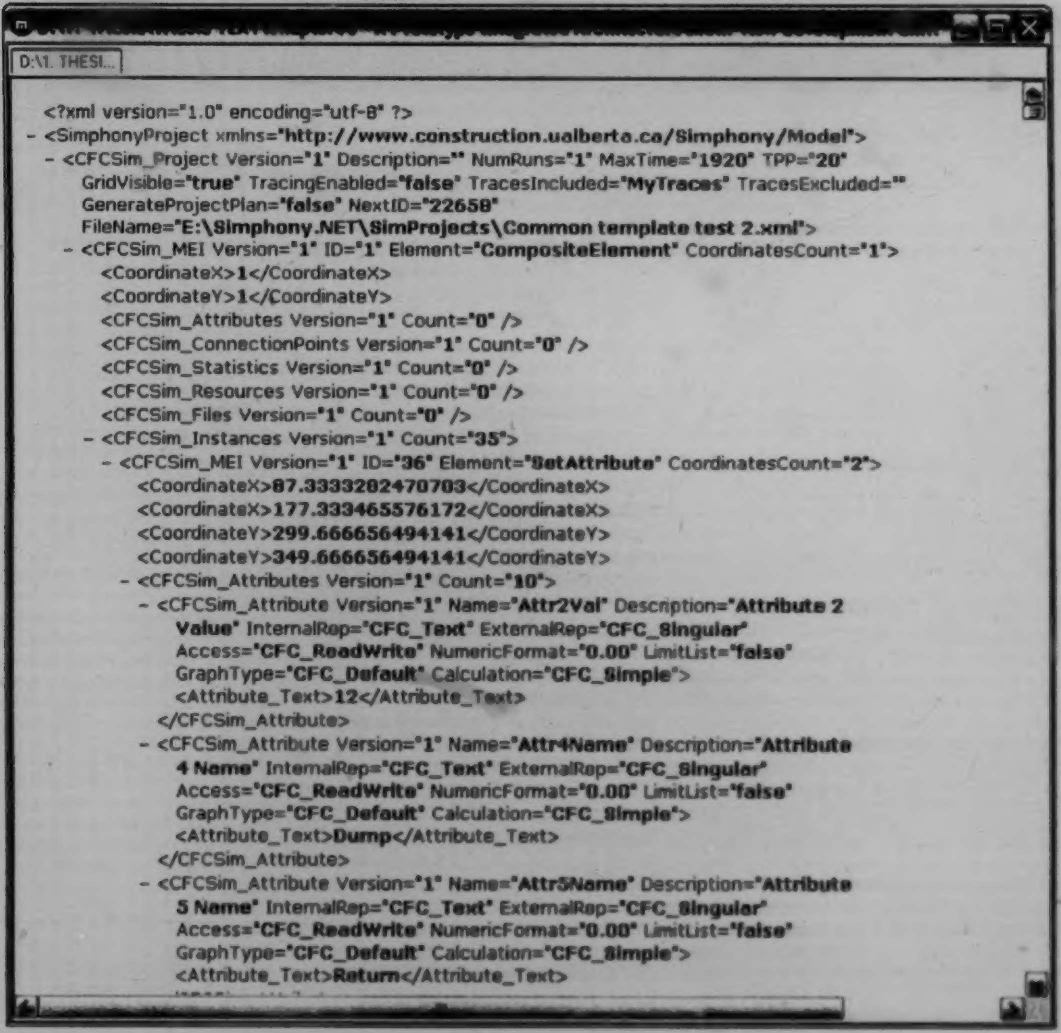
6.6 SIDE LEVEL: MULTI-VIEW DEVELOPMENT / DATA MANIPULATION

The side level is not one of the four main levels of this architecture. It is a side functional component associated with the development. This advantage is mainly owing to the XML. In the proposed architecture, the knowledge library is stored in the XML format and the developed simulation models will be saved in the XML format as well.

The benefits of XML to simulation have been mentioned several times in Chapters 5 and 6. As discussed in Section 5.6.7, the following three different views can be developed for model development or data manipulation:

- Graphic model development view
- XML text view
- Independent input/output view based on the XML text file

Of the three, the latter two views will be important new features of the prototype architecture. An example of the XML text of a simulation model is shown in Figure 6-12.



```
  <?xml version="1.0" encoding="utf-8" ?>
- <SimphonyProject xmlns="http://www.construction.ualberta.ca/Simphony/Model">
  - <CFCSim_Project Version="1" Description="" NumRuns="1" MaxTime="1920" TPP="20"
      GridVisible="true" TracingEnabled="false" TracesIncluded="MyTraces" TracesExcluded=""
      GenerateProjectPlan="false" NextID="22658"
      FileName="E:\Simphony.NET\SimProjects\Common template test 2.xml">
    - <CFCSim_MEI Version="1" ID="1" Element="CompositeElement" CoordinatesCount="1">
        <CoordinateX>1</CoordinateX>
        <CoordinateY>1</CoordinateY>
        <CFCSim_Attributes Version="1" Count="0" />
        <CFCSim_ConnectionPoints Version="1" Count="0" />
        <CFCSim_Statistics Version="1" Count="0" />
        <CFCSim_Resources Version="1" Count="0" />
        <CFCSim_Files Version="1" Count="0" />
      - <CFCSim_Instances Version="1" Count="35">
        - <CFCSim_MEI Version="1" ID="36" Element="SetAttribute" CoordinatesCount="2">
            <CoordinateX>87.3333282470703</CoordinateX>
            <CoordinateX>177.333465576172</CoordinateX>
            <CoordinateY>299.666656494141</CoordinateY>
            <CoordinateY>349.666656494141</CoordinateY>
          - <CFCSim_Attributes Version="1" Count="10">
            - <CFCSim_Attribute Version="1" Name="Attr2Val" Description="Attribute 2
                Value" InternalRep="CFC_Text" ExternalRep="CFC_Singular"
                Access="CFC_ReadWrite" NumericFormat="0.00" LimitList="false"
                GraphType="CFC_Default" Calculation="CFC_Simple">
                <Attribute_Text>12</Attribute_Text>
              </CFCSim_Attribute>
            - <CFCSim_Attribute Version="1" Name="Attr4Name" Description="Attribute
                4 Name" InternalRep="CFC_Text" ExternalRep="CFC_Singular"
                Access="CFC_ReadWrite" NumericFormat="0.00" LimitList="false"
                GraphType="CFC_Default" Calculation="CFC_Simple">
                <Attribute_Text>Dump</Attribute_Text>
              </CFCSim_Attribute>
            - <CFCSim_Attribute Version="1" Name="Attr5Name" Description="Attribute
                5 Name" InternalRep="CFC_Text" ExternalRep="CFC_Singular"
                Access="CFC_ReadWrite" NumericFormat="0.00" LimitList="false"
                GraphType="CFC_Default" Calculation="CFC_Simple">
                <Attribute_Text>Return</Attribute_Text>
```

Figure 6-12. XML Text File of a Simulation Model

In Figure 6-13, the file can be opened by an XML editor, XMLEditPro, which can manipulate the data. It is not exactly the interface, which the third view proposes; it, however, offers the user an alternate visualization of the XML-based simulation model.

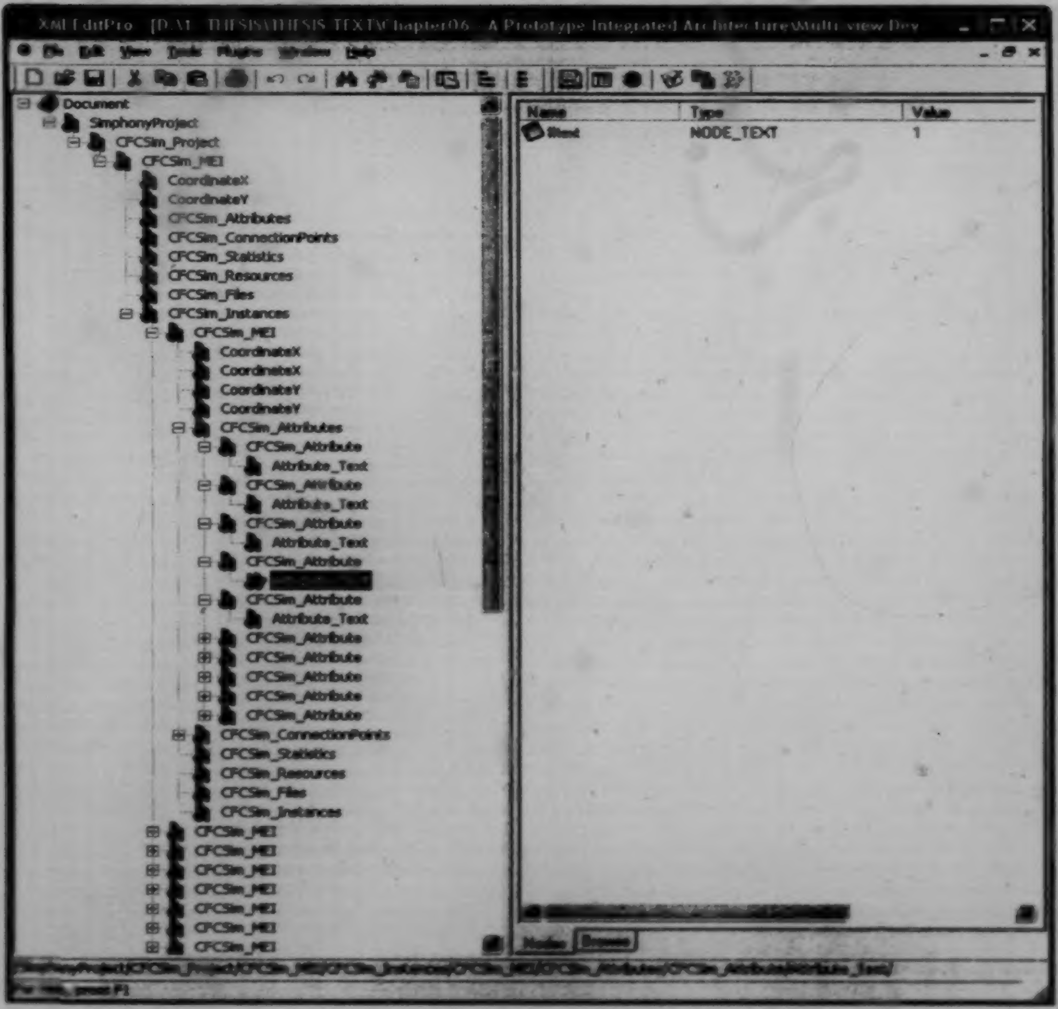

XMLEditPro - [D:\T... THESIS\THESIS TEXT\Chapter06 - A Prototype Integrated Architecture\Multi-view Dev
File Edit View Tools Plugins Window Help
Document
SimphonyProject
CFCSim_Project
CFCSim_MEI
CoordinateX
CoordinateY
CFCSim_Attributes
CFCSim_ConnectionPoints
CFCSim_Statistics
CFCSim_Resources
CFCSim_Files
CFCSim_Instances
CFCSim_MEI
CoordinateX
CoordinateX
CoordinateY
CoordinateY
CFCSim_Attributes
CFCSim_Attribute
Attribute_Text
CFCSim_Attribute
Attribute_Text
CFCSim_Attribute
Attribute_Text
CFCSim_Attribute
CFCSim_Attribute
Attribute_Text
CFCSim_Attribute
CFCSim_Attribute
CFCSim_Attribute
CFCSim_Attribute
CFCSim_Attribute
CFCSim_ConnectionPoints
CFCSim_Statistics
CFCSim_Resources
CFCSim_Files
CFCSim_Instances
CFCSim_MEI
CFCSim_MEI
CFCSim_MEI
CFCSim_MEI
CFCSim_MEI
CFCSim_MEI
CFCSim_MEI
Name
Type
Value
#text
NODE_TEXT
1
Nodes
Browser
SimphonyProject/CFCSim_Project/CFCSim_MEI/CFCSim_Instances/CFCSim_MEI/CFCSim_Attributes/CFCSim_Attribute/Attribute_Text/
For Help, press F1

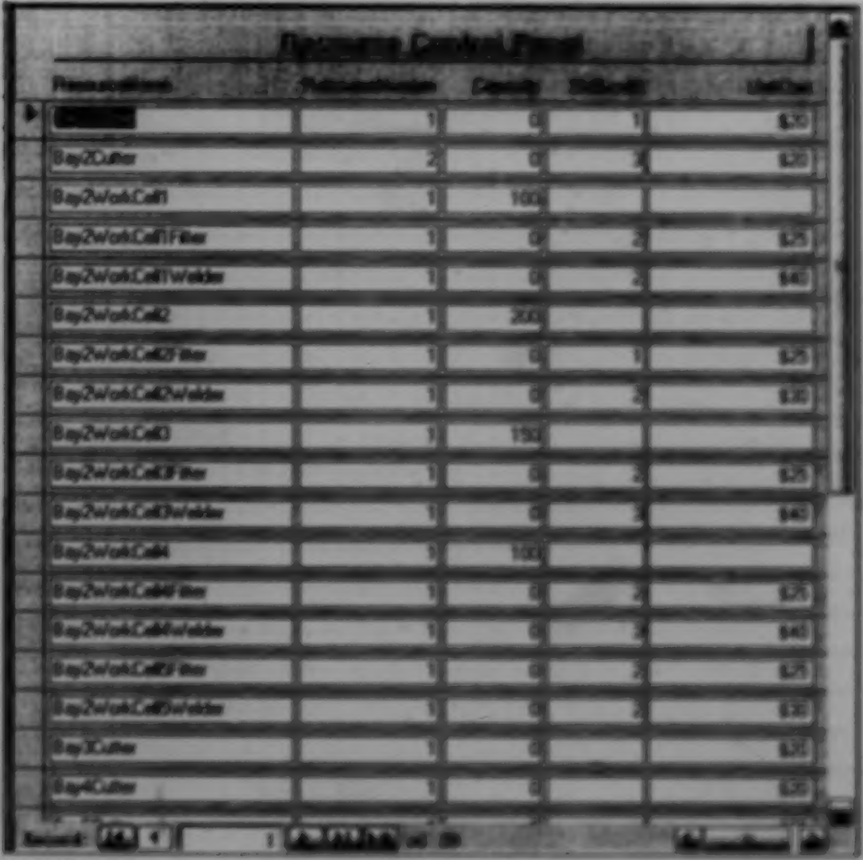
Figure 6-13. View and Manipulate a XML-based Simulation Model

The multi-view development / data manipulation serves the production-based large scale construction simulation in several ways, as explored below.

6.6.1 Increased Efficiency of Scenario Iterations

A developed simulation model is often reused with different sets of attribute data for executing the different scenario iterations. When the simulation model is large and complex, updating attribute values becomes a laborious and risky task in a graphic development interface.

An independent input/output view based on the XML text file provides us with the chance to group the information and present it in a different way. The interfaces can be programmed using an XML query language. Figure 6-14 shows a proposed control panel for all resources in a simulation model. The task of scenario iteration becomes more efficient.



Resource Control Panel

[illegible]	[illegible]	Capacity	[illegible]	UnitCost
[illegible]	1	0	1	$30
Bay2Cutter	2	0	2	$20
Bay2WorkCell1	1	100		
Bay2WorkCell1Fitter	1	0	2	$25
Bay2WorkCell1Welder	1	0	2	$40
Bay2WorkCell2	1	200		
Bay2WorkCell2Fitter	1	0	1	$25
Bay2WorkCell2Welder	1	0	2	$30
Bay2WorkCell3	1	150		
Bay2WorkCell3Fitter	1	0	2	$25
Bay2WorkCell3Welder	1	0	2	$40
Bay2WorkCell4	1	100		
Bay2WorkCell4Fitter	1	0	2	$25
Bay2WorkCell4Welder	1	0	2	$40
Bay2WorkCell5Fitter	1	0	2	$25
Bay2WorkCell5Welder	1	0	2	$30
Bay3Cutter	1	0		$30
Bay4Cutter	1	0		$30

Figure 6-14. Resource Control Panel based on the XML Model File

6.6.2 Knowledge Acquisition from Other Standards

The CSOM is presented in XML format, which facilitates the acquisition of knowledge from existing standards or from human beings. Three sources, ifcXML, construction ontology, and expert knowledge, are explored as shown in Figure 6-15. They are described in detail in Section 6.2.3.

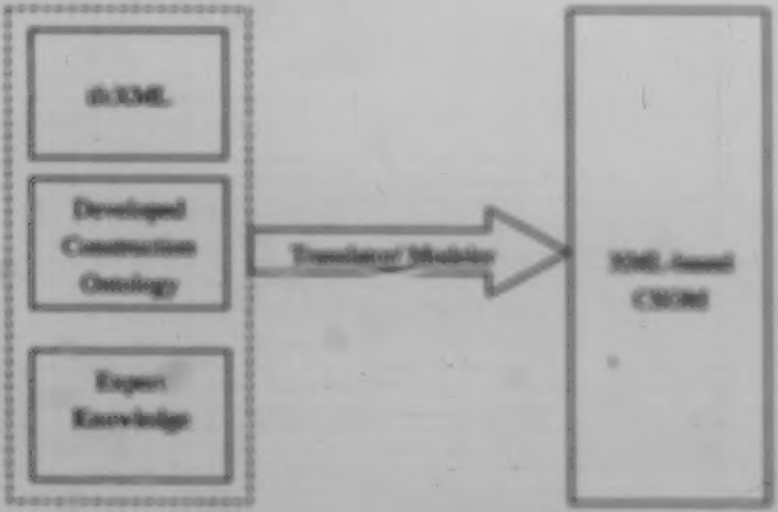

ifcXML
Developed Construction Ontology
Expert Knowledge
Translator/ Modules
XML-based CSOM

Figure 6-15. Knowledge Acquisition

6.6.3 Data Exchange with Other Applications

The design and simulation of a construction system is based on the real data drawn from existing construction applications. The data is often scattered through various sources such as computer-generated drawings, ERP applications, estimating applications, material management applications, and scheduling applications on different computers in an enterprise. The XML-based construction simulation model file eases the data exchange between a simulation model and other applications as illustrated in Figure 6-16.

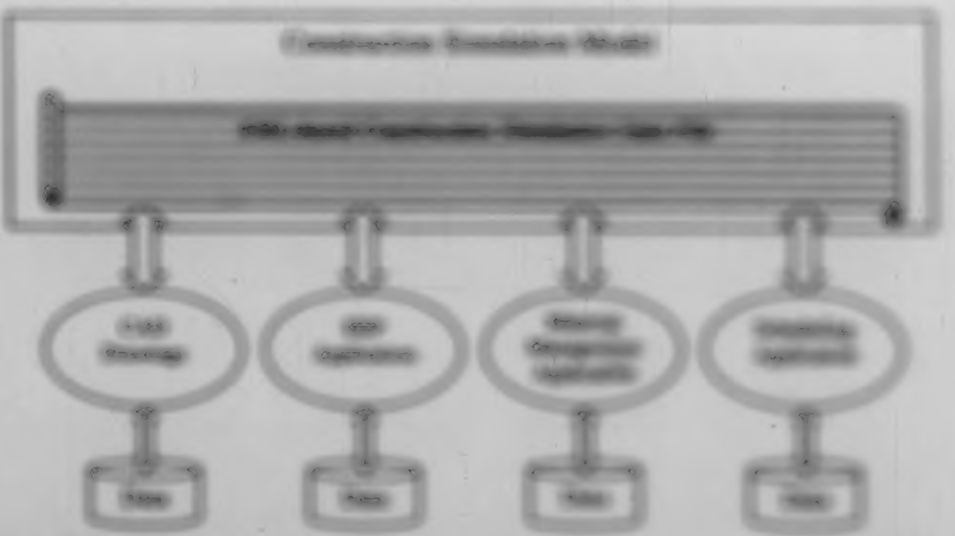

Construction Simulation Model
CAD Drawings
ERP Application
Material Management Application
Scheduling Application
Data
Data
Data
Data

Figure 6-16: Data Exchange between Simulation Models and Other Applications

6.6.5 Output Utilization and Display

Currently, simulation output is mainly used to analyze resource utilization, project duration, and productivity. For some projects, however, information can be collected and used for other purposes.

In a fabrication shop, the start and finish time of every process of each spool can be collected. When the output is stored in the XML model file as well, they can be easily retrieved, and even automatically imported to a scheduling system, facilitating shop scheduling. The XML-based output data can also be retrieved and displayed on Web pages.

6.7 CASE STUDY

Much knowledge of industrial construction was gained and accumulated during phase one research. Therefore, an industrial construction project was chosen as the case study to demonstrate the simulation modeling procedure using the proposed architecture. The model is simplified to some extent due to the unavailability of the full development of the proposed architecture. As such, it does not include all features of production-based large scale construction simulation. It does demonstrate the basic important procedures of the designed prototype architecture with a focus on building an HLA simulation federation.

6.7.1 Knowledge Acquisition, Standardization, and Library

In phase one research, the author modeled spool fabrication and the entire industrial construction project system. In this chapter, the author collected and modeled the knowledge of industrial construction projects, and stored them in the structured knowledge library. A complete CSOM for the industrial construction domain is developed in this case study for simulation of industrial construction. Figure 6-17 is the UML static model for the developed CSOM. Figure 6-18 shows the built CSOM libraries viewed from the object model editor.

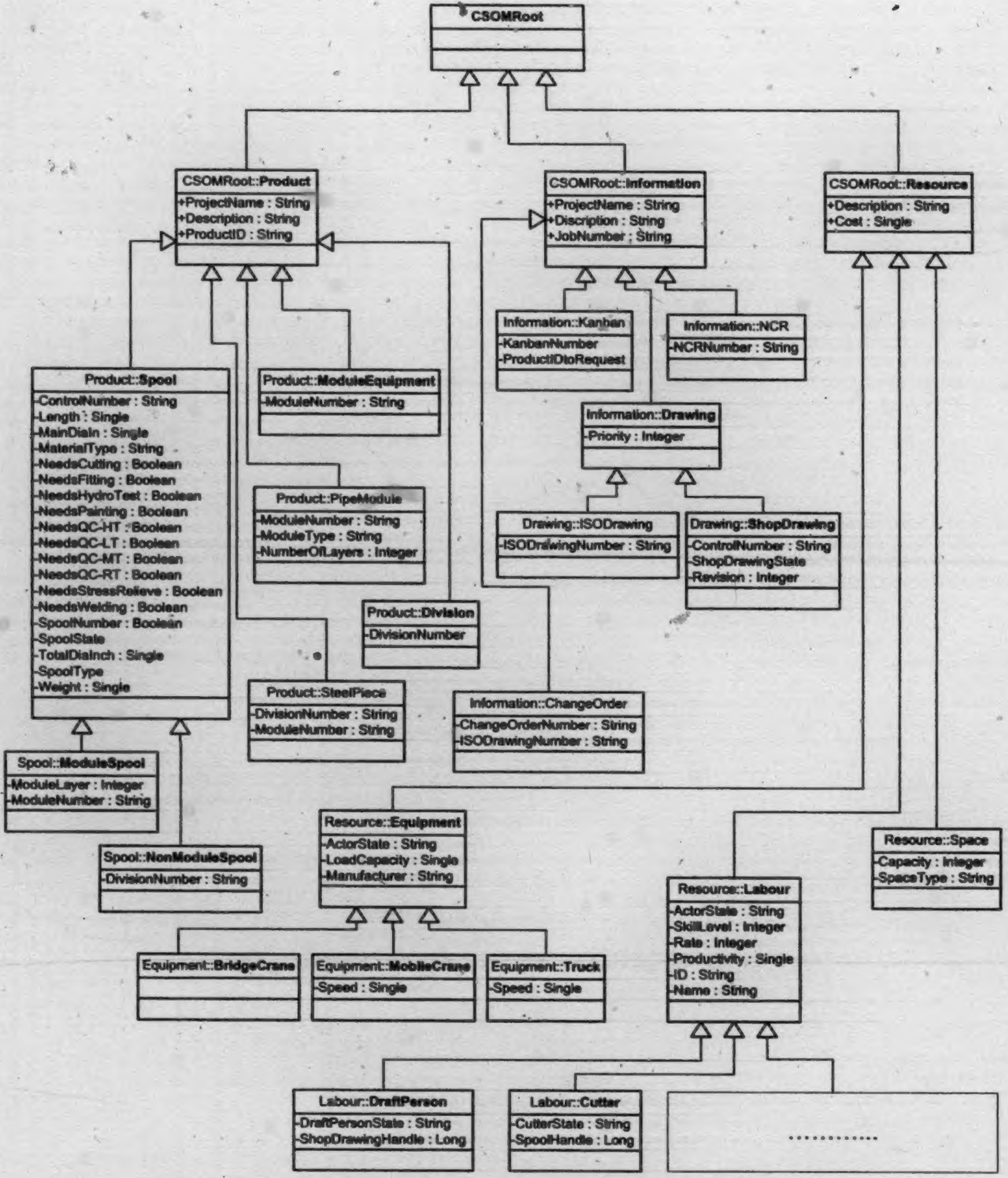
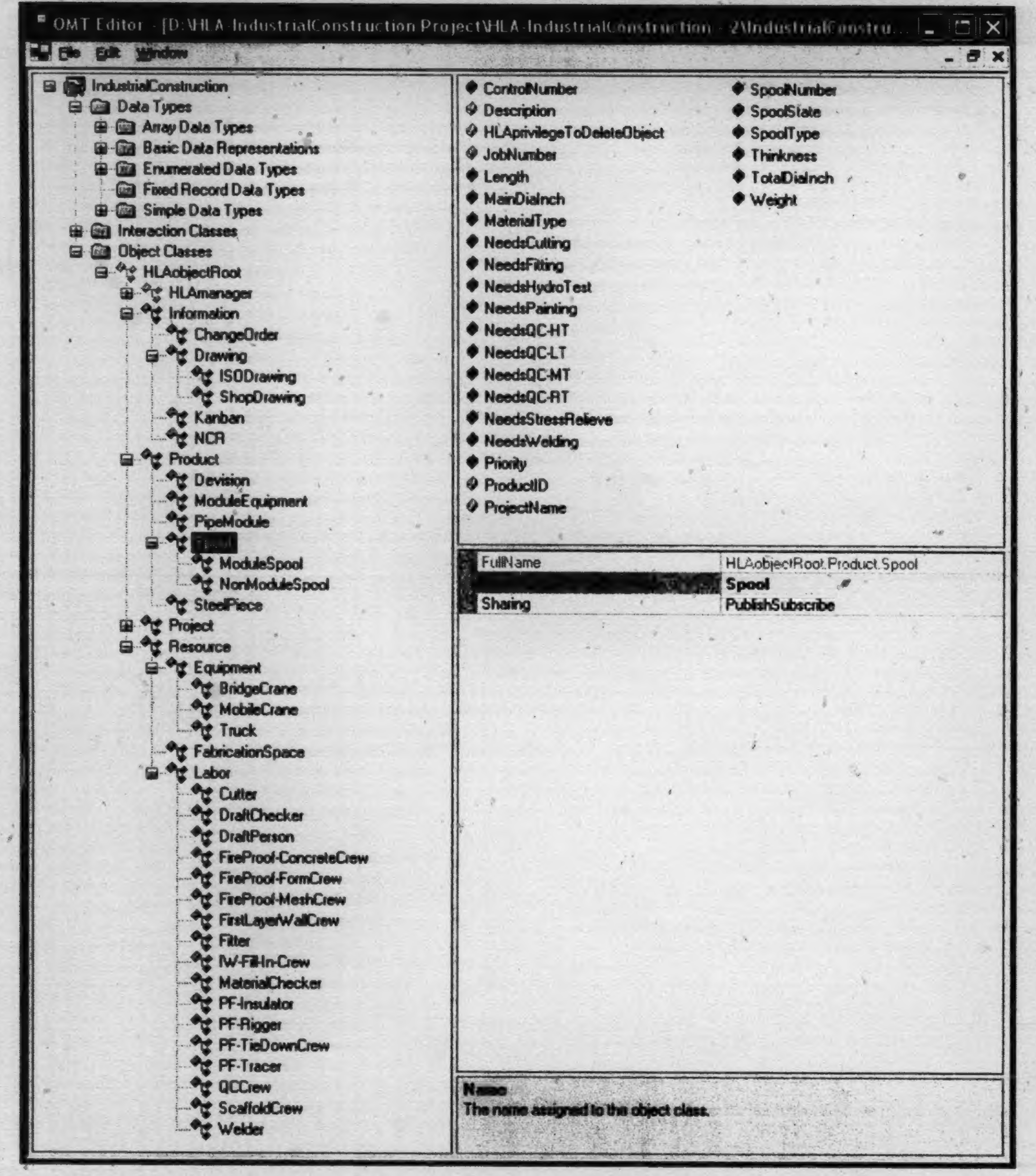

CSOMRoot
CSOMRoot::Product
+ProjectName : String
+Description : String
+ProductID : String
CSOMRoot::Information
+ProjectName : String
+Discription : String
+JobNumber : String
CSOMRoot::Resource
+Description : String
+Cost : Single
Information::Kanban
-KanbanNumber
-ProductIDtoRequest
Information::NCR
-NCRNumber : String
Product::Spool
-ControlNumber : String
-Length : Single
-MainDiaIn : Single
-MaterialType : String
-NeedsCutting : Boolean
-NeedsFitting : Boolean
-NeedsHydroTest : Boolean
-NeedsPainting : Boolean
-NeedsQC-HT : Boolean
-NeedsQC-LT : Boolean
-NeedsQC-MT : Boolean
-NeedsQC-RT : Boolean
-NeedsStressRelieve : Boolean
-NeedsWelding : Boolean
-SpoolNumber : Boolean
-SpoolState
-TotalDiaInch : Single
-SpoolType
-Weight : Single
Product::ModuleEquipment
-ModuleNumber : String
Information::Drawing
-Priority : Integer
Product::PipeModule
-ModuleNumber : String
-ModuleType : String
-NumberOfLayers : Integer
Drawing::ISODrawing
-ISODrawingNumber : String
Drawing::ShopDrawing
-ControlNumber : String
-ShopDrawingState
-Revision : Integer
Product::Division
-DivisionNumber
Product::SteelPiece
-DivisionNumber : String
-ModuleNumber : String
Information::ChangeOrder
-ChangeOrderNumber : String
-ISODrawingNumber : String
Spool::ModuleSpool
-ModuleLayer : Integer
-ModuleNumber : String
Spool::NonModuleSpool
-DivisionNumber : String
Resource::Equipment
-ActorState : String
-LoadCapacity : Single
-Manufacturer : String
Resource::Space
-Capacity : Integer
-SpaceType : String
Resource::Labour
-ActorState : String
-SkillLevel : Integer
-Rate : Integer
-Productivity : Single
-ID : String
-Name : String
Equipment::BridgeCrane
Equipment::MobileCrane
-Speed : Single
Equipment::Truck
-Speed : Single
Labour::DraftPerson
-DraftPersonState : String
-ShopDrawingHandle : Long
Labour::Cutter
-CutterState : String
-SpoolHandle : Long
....

Figure 6-17. UML Static Model for Industrial Construction CSOM


OMT Editor - [D:\HLA-IndustrialConstruction Project\HLA-IndustrialConstruction - 2\IndustrialConstru...
File
Edit
Window
IndustrialConstruction
Data Types
Array Data Types
Basic Data Representations
Enumerated Data Types
Fixed Record Data Types
Simple Data Types
Interaction Classes
Object Classes
HLAobjectRoot
HLAmanager
Information
ChangeOrder
Drawing
ISODrawing
ShopDrawing
Kanban
NCR
Product
Devision
ModuleEquipment
PipeModule
Spool
ModuleSpool
NonModuleSpool
SteelPiece
Project
Resource
Equipment
BridgeCrane
MobileCrane
Truck
FabricationSpace
Labor
Cutter
DraftChecker
DraftPerson
FireProof-ConcreteCrew
FireProof-FormCrew
FireProof-MeshCrew
FirstLayerWallCrew
Fitter
IW-FillIn-Crew
MaterialChecker
PF-Insulator
PF-Rigger
PF-TieDownCrew
PF-Tracer
QCCrew
ScaffoldCrew
Welder
ControlNumber
Description
HLAprivilegeToDeleteObject
JobNumber
Length
MainDiaInch
MaterialType
NeedsCutting
NeedsFitting
NeedsHydroTest
NeedsPainting
NeedsQC-HT
NeedsQC-LT
NeedsQC-MT
NeedsQC-RT
NeedsStressRelieve
NeedsWelding
Priority
ProductID
ProjectName
SpoolNumber
SpoolState
SpoolType
Thinkness
TotalDiaInch
Weight

FullName	HLAobjectRoot.Product.Spool
	Spool
Sharing	PublishSubscribe

Name
The name assigned to the object class.

Figure 6-18. A Complete CSOM for the Industrial Construction Project

6.7.2 Generate a Specific eFOM

A specific eFOM needs to be generated as a subset from the above CSOM library of industrial construction project. In this case study, since the author gained and developed all the knowledge by himself and for the sole purpose of developing a simulation model, the generated eFOM is same as the CSOM library.

6.7.3 Design and Develop a Simulation Federation

Due to the unavailability of a graphic developing interface, the simulation model was developed directly using VB.NET with the developed modeling element classes, simulation services, and HLA-related services. As such, it is not as complex as the model developed in Chapter 3. It is a simplified version of that model within the new framework. The federation is named "IndustrialConstruction", and is decomposed into 4 federates:

- Drafting Federate,
- Procurement Federate,
- Fabrication Federate,
- Module Assembly Federate.

The four federates are developed separately. Figures 6-19 through 6-22 depict the flow diagrams of the algorithm for the four federates.

In the drafting federate (Figure 6-19), 100 shop drawings are created as the instances of the "ShopDrawing" class from the eFOM object library by the *SpontaneousCreatorElement*. Resources are defined from the object library as well. Each created shop drawing entity flows to *CaptureElement*, which is triggered to capture a Draftperson resource. The *SetOutputTimeElement* defines the duration of the drafting process. After the defined time interval, the state type of the shop drawing is updated to "Drafted" by the *UpdateElement*. The state type is the new feature of the HLA-compliant simulation. All possible state types of an object are predefined in the object library. In a similar way, shop drawings are checked, redrafted, rechecked, and material checked. Finally, the state type of a shop drawing is updated to "MaterialChecked". Afterward, the drafting federate gives up the ownership of the shop drawing entity through *DivestElement*. This action signifies that the drafting federate loses the authority to change the attribute values of the shop drawing.

In the procurement federate (Figure 6-20), the *DiscoverElement* discovers the creation of shop drawings and the *ObserverElement* monitors the state type of shop drawings. When the state type of a shop drawing becomes "MaterialChecked", the *TriggeredQueueElement* is triggered and adds the shop drawing entity to the Procurement_Queue. The *AcquireElement* gets the ownership of the shop drawing, which means that it gets the authority to change the attribute values of the shop drawing. The entity then triggers the procurement process. After that, its state type is updated to "MaterialBeingProcured". Thereafter, the procurement federate gives up the ownership of the shop drawing entity through *DivestElement*.

The *ObserverElement* in the fabrication federate (Figure 6-21) notes that the state type of shop drawings becomes "MaterialBeingProcured". It then triggers the *TriggeredQueueElement* together with the *DiscoverElement* to add the shop drawing to the Fabrication_Queue. Each shop drawing entity is next transformed into a spool entity through the *TransformCreatorElement*, in which a spool is created as an object instance of the "Spool" class from the eFOM object library. Thereafter, the spool is cut, fitted, welded, quality checked, and painted using different resources. Its state type is updated after each process; the final state type is "Painted". In the end, the fabrication federate loses the ownership of the spool through *DivestElement*.

When the *ObserverElement* in the module assembly federate (Figure 6-22) observes that the state type of a spool becomes "Painted", it then triggers the *TriggeredQueueElement* together with the *DiscoverElement* to add the spool to the ModuleAssembly_Queue. It is assumed in this case study that every twenty spools are assembled for one module and that they are fabricated in the correct sequence. As such, twenty spool entities are consolidated into one entity by the *ConsolidateElement*. Each entity is then transformed into a module entity through the *TransformCreatorElement*, in which a module is created as an object instance of the "PipeModule" class from the eFOM object library. The module is assembled and its state type is updated to "Assembled". The program ends when the assembled module has reached "five" in the *CounterElement*.

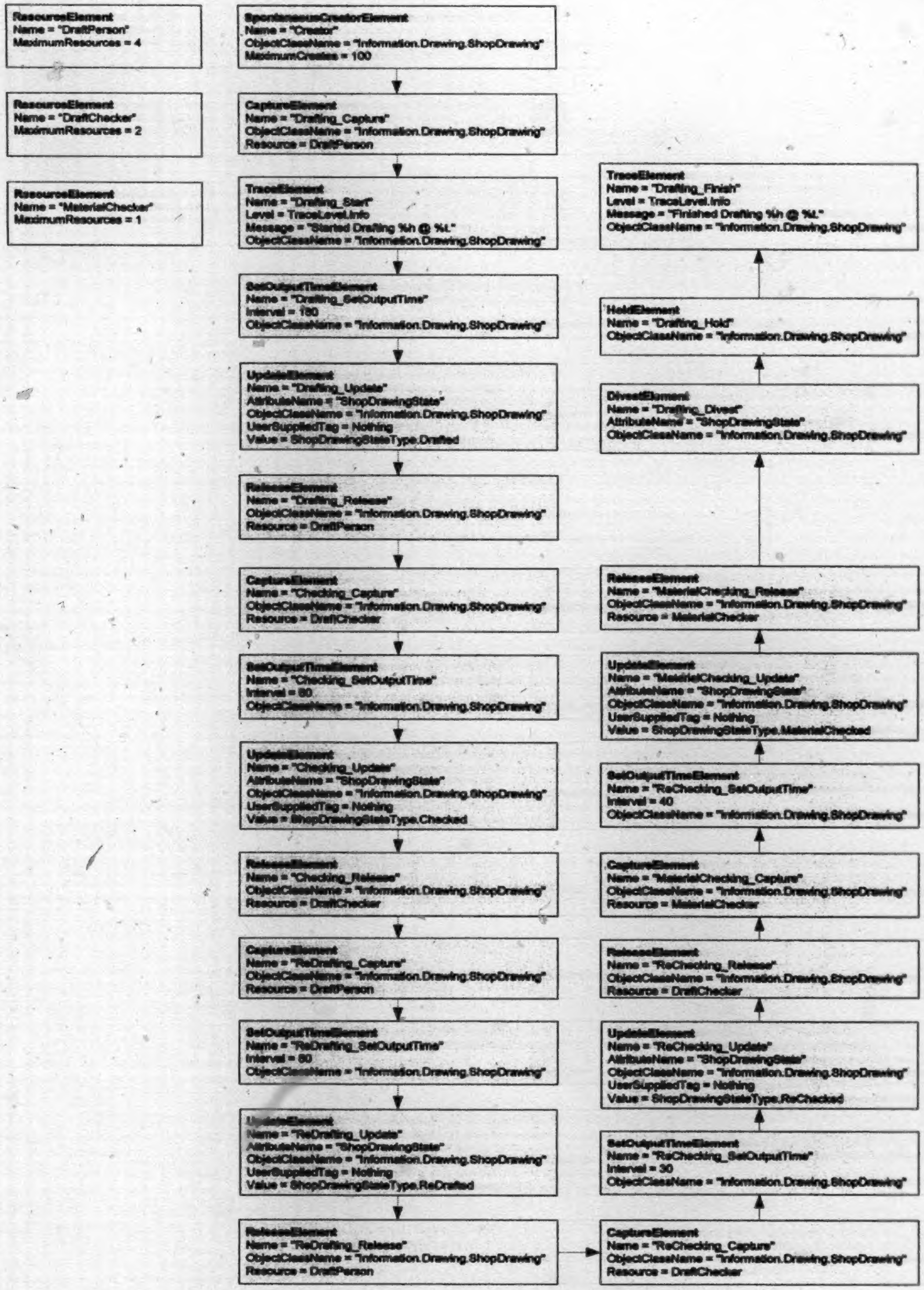

ResourceElement
Name = "DraftPerson"
MaximumResources = 4
ResourceElement
Name = "DraftChecker"
MaximumResources = 2
ResourceElement
Name = "MaterialChecker"
MaximumResources = 1
SpontaneousCreatorElement
Name = "Creator"
ObjectClassName = "Information.Drawing.ShopDrawing"
MaximumCreates = 100
CaptureElement
Name = "Drafting_Capture"
ObjectClassName = "Information.Drawing.ShopDrawing"
Resource = DraftPerson
TraceElement
Name = "Drafting_Start"
Level = TraceLevel.Info
Message = "Started Drafting %h @ %t."
ObjectClassName = "Information.Drawing.ShopDrawing"
SetOutputTimeElement
Name = "Drafting_SetOutputTime"
Interval = 180
ObjectClassName = "Information.Drawing.ShopDrawing"
UpdateElement
Name = "Drafting_Update"
AttributeName = "ShopDrawingState"
ObjectClassName = "Information.Drawing.ShopDrawing"
UserSuppliedTag = Nothing
Value = ShopDrawingStateType.Drafted
ReleaseElement
Name = "Drafting_Release"
ObjectClassName = "Information.Drawing.ShopDrawing"
Resource = DraftPerson
CaptureElement
Name = "Checking_Capture"
ObjectClassName = "Information.Drawing.ShopDrawing"
Resource = DraftChecker
SetOutputTimeElement
Name = "Checking_SetOutputTime"
Interval = 60
ObjectClassName = "Information.Drawing.ShopDrawing"
UpdateElement
Name = "Checking_Update"
AttributeName = "ShopDrawingState"
ObjectClassName = "Information.Drawing.ShopDrawing"
UserSuppliedTag = Nothing
Value = ShopDrawingStateType.Checked
ReleaseElement
Name = "Checking_Release"
ObjectClassName = "Information.Drawing.ShopDrawing"
Resource = DraftChecker
CaptureElement
Name = "ReDrafting_Capture"
ObjectClassName = "Information.Drawing.ShopDrawing"
Resource = DraftPerson
SetOutputTimeElement
Name = "ReDrafting_SetOutputTime"
Interval = 60
ObjectClassName = "Information.Drawing.ShopDrawing"
UpdateElement
Name = "ReDrafting_Update"
AttributeName = "ShopDrawingState"
ObjectClassName = "Information.Drawing.ShopDrawing"
UserSuppliedTag = Nothing
Value = ShopDrawingStateType.ReDrafted
ReleaseElement
Name = "ReDrafting_Release"
ObjectClassName = "Information.Drawing.ShopDrawing"
Resource = DraftPerson
CaptureElement
Name = "ReChecking_Capture"
ObjectClassName = "Information.Drawing.ShopDrawing"
Resource = DraftChecker
SetOutputTimeElement
Name = "ReChecking_SetOutputTime"
Interval = 30
ObjectClassName = "Information.Drawing.ShopDrawing"
UpdateElement
Name = "ReChecking_Update"
AttributeName = "ShopDrawingState"
ObjectClassName = "Information.Drawing.ShopDrawing"
UserSuppliedTag = Nothing
Value = ShopDrawingStateType.ReChecked
ReleaseElement
Name = "ReChecking_Release"
ObjectClassName = "Information.Drawing.ShopDrawing"
Resource = DraftChecker
CaptureElement
Name = "MaterialChecking_Capture"
ObjectClassName = "Information.Drawing.ShopDrawing"
Resource = MaterialChecker
SetOutputTimeElement
Name = "ReChecking_SetOutputTime"
Interval = 40
ObjectClassName = "Information.Drawing.ShopDrawing"
UpdateElement
Name = "MaterialChecking_Update"
AttributeName = "ShopDrawingState"
ObjectClassName = "Information.Drawing.ShopDrawing"
UserSuppliedTag = Nothing
Value = ShopDrawingStateType.MaterialChecked
ReleaseElement
Name = "MaterialChecking_Release"
ObjectClassName = "Information.Drawing.ShopDrawing"
Resource = MaterialChecker
DivestElement
Name = "Drafting_Divest"
AttributeName = "ShopDrawingState"
ObjectClassName = "Information.Drawing.ShopDrawing"
HoldElement
Name = "Drafting_Hold"
ObjectClassName = "Information.Drawing.ShopDrawing"
TraceElement
Name = "Drafting_Finish"
Level = TraceLevel.Info
Message = "Finished Drafting %h @ %t."
ObjectClassName = "Information.Drawing.ShopDrawing"

Figure 6-19. Flow Diagram of Drafting Federate

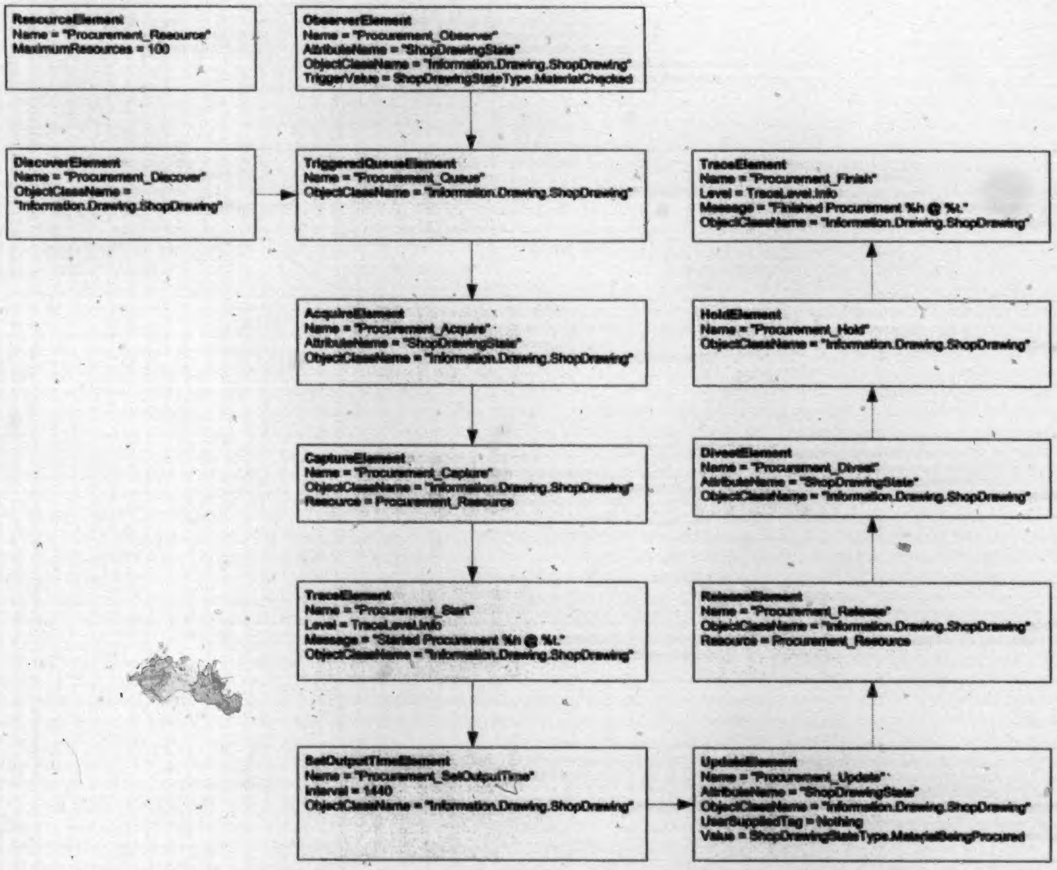

ResourceElement
Name = "Procurement_Resource"
MaximumResources = 100
ObserverElement
Name = "Procurement_Observer"
AttributeName = "ShopDrawingState"
ObjectClassName = "Information.Drawing.ShopDrawing"
TriggerValue = ShopDrawingStateType.MaterialChecked
DiscoverElement
Name = "Procurement_Discover"
ObjectClassName =
"Information.Drawing.ShopDrawing"
TriggeredQueueElement
Name = "Procurement_Queue"
ObjectClassName = "Information.Drawing.ShopDrawing"
TraceElement
Name = "Procurement_Finish"
Level = TraceLevel.Info
Message = "Finished Procurement %h @ %t."
ObjectClassName = "Information.Drawing.ShopDrawing"
AcquireElement
Name = "Procurement_Acquire"
AttributeName = "ShopDrawingState"
ObjectClassName = "Information.Drawing.ShopDrawing"
HoldElement
Name = "Procurement_Hold"
ObjectClassName = "Information.Drawing.ShopDrawing"
CaptureElement
Name = "Procurement_Capture"
ObjectClassName = "Information.Drawing.ShopDrawing"
Resource = Procurement_Resource
DivestElement
Name = "Procurement_Divest"
AttributeName = "ShopDrawingState"
ObjectClassName = "Information.Drawing.ShopDrawing"
TraceElement
Name = "Procurement_Start"
Level = TraceLevel.Info
Message = "Started Procurement %h @ %t."
ObjectClassName = "Information.Drawing.ShopDrawing"
ReleaseElement
Name = "Procurement_Release"
ObjectClassName = "Information.Drawing.ShopDrawing"
Resource = Procurement_Resource
SetOutputTimeElement
Name = "Procurement_SetOutputTime"
Interval = 1440
ObjectClassName = "Information.Drawing.ShopDrawing"
UpdateElement
Name = "Procurement_Update"
AttributeName = "ShopDrawingState"
ObjectClassName = "Information.Drawing.ShopDrawing"
UserSuppliedTag = Nothing
Value = ShopDrawingStateType.MaterialBeingProcured

Figure 6-20. Flow Diagram of Procurement Federate

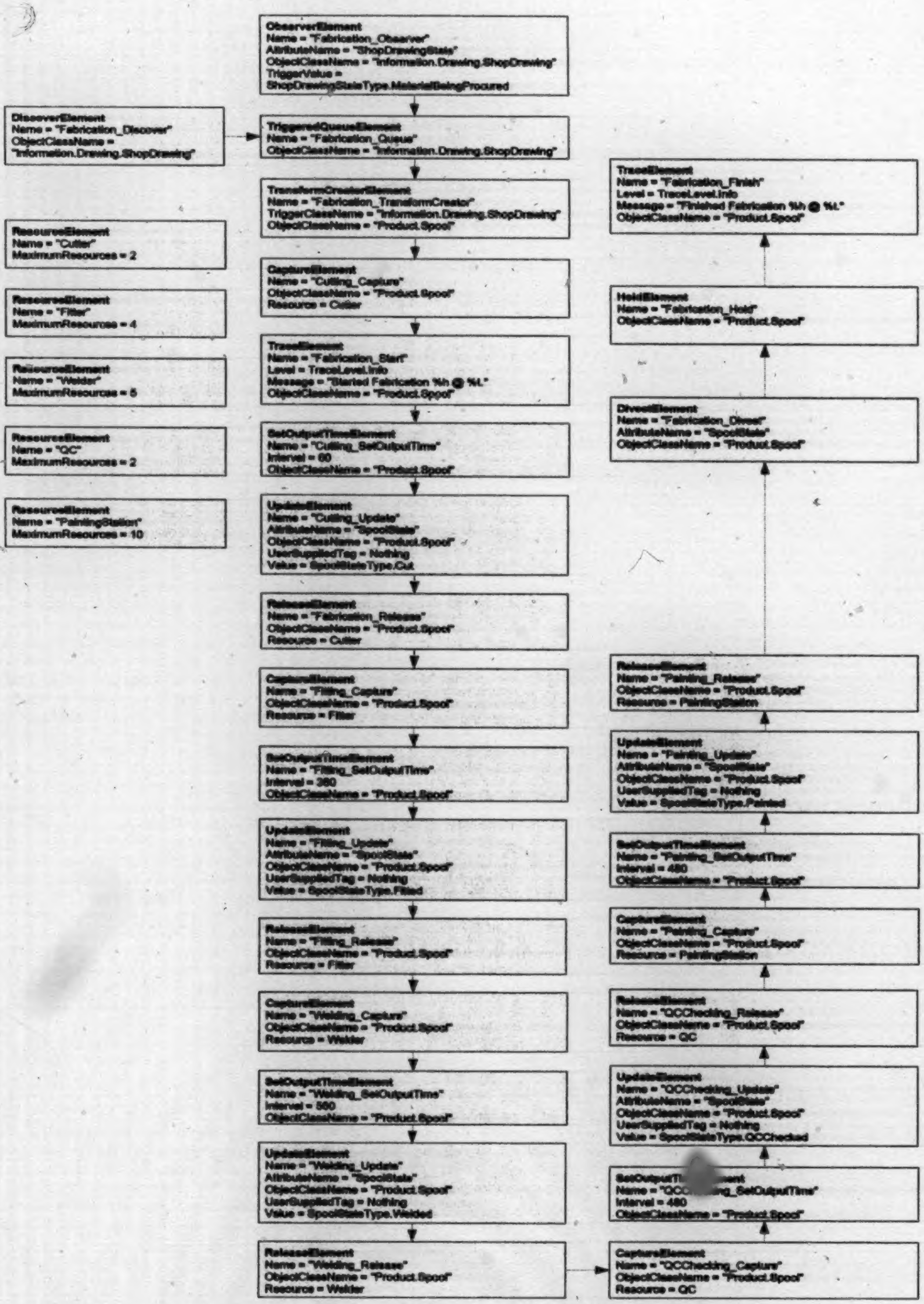

ObserverElement
Name = "Fabrication_Observer"
AttributeName = "ShopDrawingState"
ObjectClassName = "Information.Drawing.ShopDrawing"
TriggerValue =
ShopDrawingStateType.MaterialBeingProcured
DiscoverElement
Name = "Fabrication_Discover"
ObjectClassName =
"Information.Drawing.ShopDrawing"
TriggeredQueueElement
Name = "Fabrication_Queue"
ObjectClassName = "Information.Drawing.ShopDrawing"
TransformCreatorElement
Name = "Fabrication_TransformCreator"
TriggerClassName = "Information.Drawing.ShopDrawing"
ObjectClassName = "Product.Spool"
ResourceElement
Name = "Cutter"
MaximumResources = 2
ResourceElement
Name = "Fitter"
MaximumResources = 4
ResourceElement
Name = "Welder"
MaximumResources = 5
ResourceElement
Name = "QC"
MaximumResources = 2
ResourceElement
Name = "PaintingStation"
MaximumResources = 10
CaptureElement
Name = "Cutting_Capture"
ObjectClassName = "Product.Spool"
Resource = Cutter
TraceElement
Name = "Fabrication_Start"
Level = TraceLevel.Info
Message = "Started Fabrication %h @ %L"
ObjectClassName = "Product.Spool"
SetOutputTimeElement
Name = "Cutting_SetOutputTime"
Interval = 60
ObjectClassName = "Product.Spool"
UpdateElement
Name = "Cutting_Update"
AttributeName = "SpoolState"
ObjectClassName = "Product.Spool"
UserSuppliedTag = Nothing
Value = SpoolStateType.Cut
ReleaseElement
Name = "Fabrication_Release"
ObjectClassName = "Product.Spool"
Resource = Cutter
CaptureElement
Name = "Fitting_Capture"
ObjectClassName = "Product.Spool"
Resource = Fitter
SetOutputTimeElement
Name = "Fitting_SetOutputTime"
Interval = 360
ObjectClassName = "Product.Spool"
UpdateElement
Name = "Fitting_Update"
AttributeName = "SpoolState"
ObjectClassName = "Product.Spool"
UserSuppliedTag = Nothing
Value = SpoolStateType.Fitted
ReleaseElement
Name = "Fitting_Release"
ObjectClassName = "Product.Spool"
Resource = Fitter
CaptureElement
Name = "Welding_Capture"
ObjectClassName = "Product.Spool"
Resource = Welder
SetOutputTimeElement
Name = "Welding_SetOutputTime"
Interval = 560
ObjectClassName = "Product.Spool"
UpdateElement
Name = "Welding_Update"
AttributeName = "SpoolState"
ObjectClassName = "Product.Spool"
UserSuppliedTag = Nothing
Value = SpoolStateType.Welded
ReleaseElement
Name = "Welding_Release"
ObjectClassName = "Product.Spool"
Resource = Welder
CaptureElement
Name = "QCChecking_Capture"
ObjectClassName = "Product.Spool"
Resource = QC
SetOutputTimeElement
Name = "QCChecking_SetOutputTime"
Interval = 480
ObjectClassName = "Product.Spool"
UpdateElement
Name = "QCChecking_Update"
AttributeName = "SpoolState"
ObjectClassName = "Product.Spool"
UserSuppliedTag = Nothing
Value = SpoolStateType.QCChecked
ReleaseElement
Name = "QCChecking_Release"
ObjectClassName = "Product.Spool"
Resource = QC
CaptureElement
Name = "Painting_Capture"
ObjectClassName = "Product.Spool"
Resource = PaintingStation
SetOutputTimeElement
Name = "Painting_SetOutputTime"
Interval = 480
ObjectClassName = "Product.Spool"
UpdateElement
Name = "Painting_Update"
AttributeName = "SpoolState"
ObjectClassName = "Product.Spool"
UserSuppliedTag = Nothing
Value = SpoolStateType.Painted
ReleaseElement
Name = "Painting_Release"
ObjectClassName = "Product.Spool"
Resource = PaintingStation
DivestElement
Name = "Fabrication_Divest"
AttributeName = "SpoolState"
ObjectClassName = "Product.Spool"
HoldElement
Name = "Fabrication_Hold"
ObjectClassName = "Product.Spool"
TraceElement
Name = "Fabrication_Finish"
Level = TraceLevel.Info
Message = "Finished Fabrication %h @ %L"
ObjectClassName = "Product.Spool"

Figure 6-21. Flow Diagram of Fabrication Federate

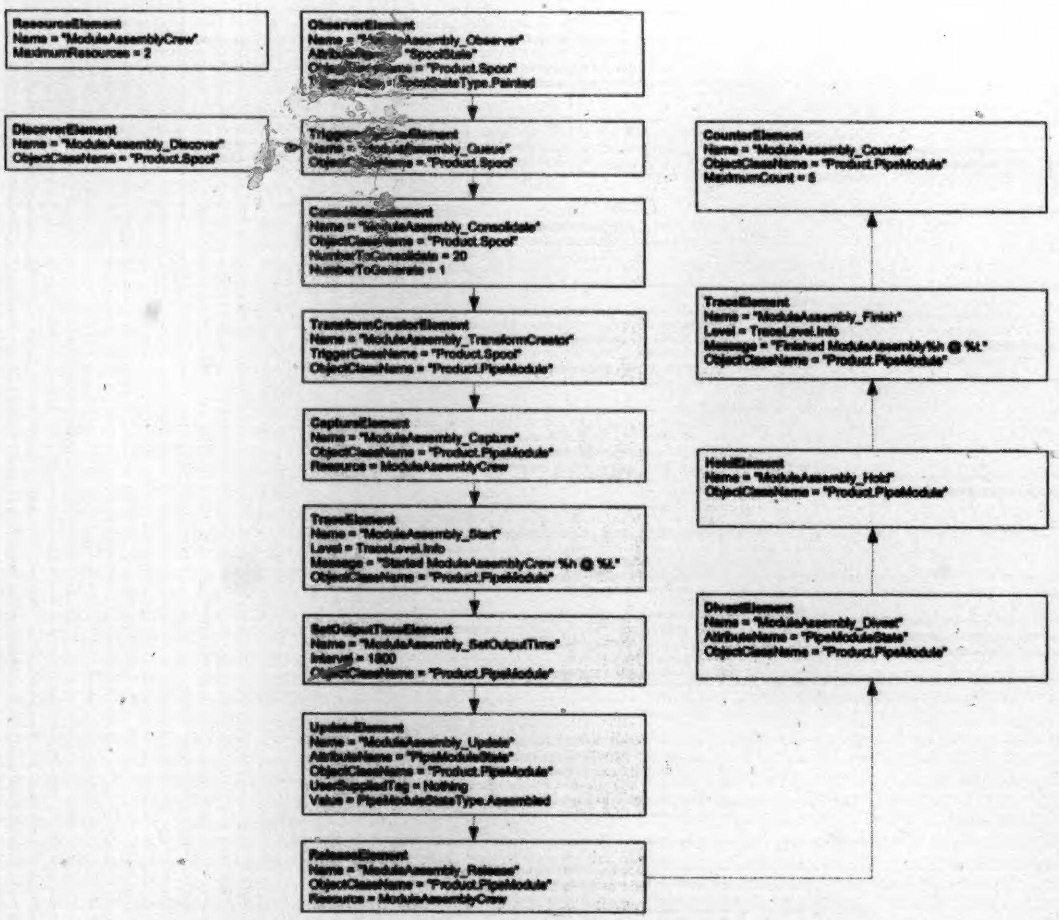


Figure 6-22. Flow Diagram of ModuleAssembly Federate

The four developed federates are constructed into a federation, which communicates through RTI. Figure 6-23 shows the entire structure of the "IndustrialConstruction" federation.

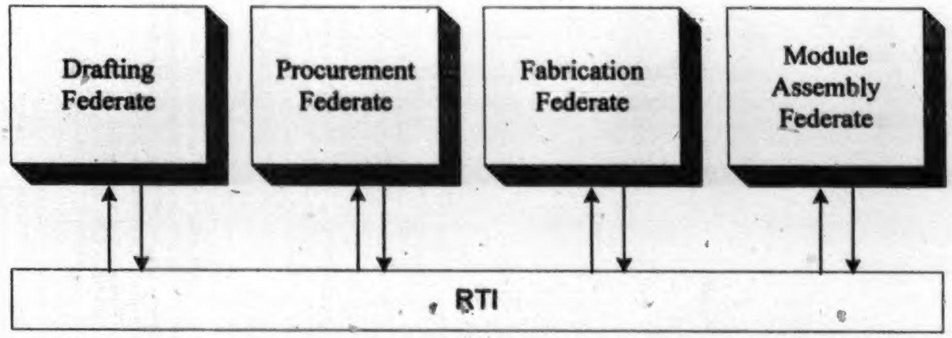


Figure 6-23. Flow Chart of Federation "IndustrialConstruction"

6.7.4 Demonstration of the Developed Federation

The developed federation is run on two computers in order to demonstrate the distributed simulation. The Drafting Federate and the Procurement Federate are located and run on computer A; the Fabrication Federate and the Module Assembly Federate are located and run on computer B. The RTI is located on the server, which is another computer. Simple interfaces are developed for each of the four federates to provide commands for creating a federation, joining a federation, executing the federate, and resigning the federate from a federation. The interface also shows both execution information and tracing information when the federation is executed. Figure 6-24 shows the screenshots of computer A and computer B when the "IndustrialConstruction" federation is executed.

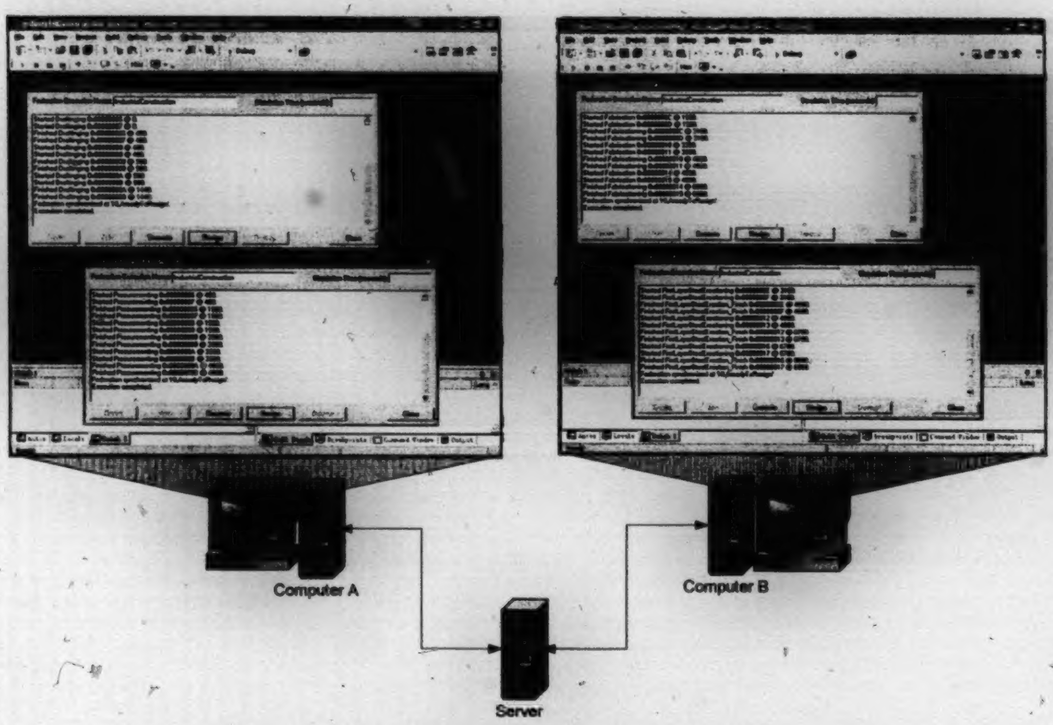


Figure 6-24. Demonstration of Execution of the Federation

6.8 CONCLUSIONS

A prototype integrated architecture for production-based large scale construction simulations is presented in this chapter. This architecture integrates solutions for identified challenges into one framework. It increases knowledge standardization and reuse, model decomposability, computing ability, product representation, model openness, and views of model development/data manipulation. It subsequently provides the foundation for increasing the capacity of large scale construction simulation models and for improving the cost-effectiveness of its development.

The proposed integrated architecture is currently a prototype. The whole system implementation is beyond the scope of the thesis. The main contribution of this chapter is that the author clearly describes and discusses the essential issues of the integrated architecture necessary for production-based large scale construction simulation modeling.

Although the implementation is not complete, some components of the framework have been developed. The basic simulation services, RTI, and some modeling elements have been developed by the NSERC/Alberta Construction Industry Research Chair simulation team. The author designed the knowledge structure and the means of knowledge acquisition and standardization. Knowledge of industrial construction has been acquired and stored in the CSOM library. A case study to implement the proposed architecture is developed in a simpler version. The CSOM library for the industrial construction domain is maintained throughout the development and can be used by future developers. The developed four federates can also be reused to develop new simulation models in this domain.

6.9 REFERENCES

El-Diraby, T. E., and Briceno, F. (2005). "Taxonomy for Outside Plant Construction in Telecommunication Infrastructure: Supporting Knowledge-Based Virtual Teaming." *Journal of Infrastructure Systems*, Vol. 11, No. 2, pp. 110-121.

El-Diraby, T. E., and Kashif, K. F. (2005). "Distributed Ontology Architecture for Knowledge Management in Highway Construction." *Journal of Construction Engineering and Management*, Vol. 131, No. 5, pp. 591-603.

Hague, S. (2005). *Simphony HLA Framework*, Internal Report, Construction Engineering and Management Group, University of Alberta.

Nisbet, N., and Liebich, T. (2005). "ifcXML Implementation Guide." <http://www.iai-international.org/Model/IfcXML2.htm> (Jun. 8, 2005).

NSERC/Alberta Construction Industry Research Chair. (2005). "Simphony HLA [online]". < http://irc.construction.ualberta.ca/simphony2/>. Accessed November 1, 2005.

CHAPTER 7 – FINAL DISCUSSION

7.1 RESEARCH SUMMARY

Initially, this thesis reviewed a large volume of literature regarding construction management theories, modeling tools, and construction simulation. The role and purpose of production-based large scale simulation were clarified. The concept of production-based large scale construction simulation was also defined.

In order to extend the use of construction simulation and demonstrate the usefulness of production-based large scale construction simulation in construction management, the author completed two projects that aimed to apply construction simulation in industrial construction. A simulation-based approach was designed and used to facilitate the implementation of lean production techniques in spool fabrication shops. It was proven a useful and more powerful approach. The research then extended to an entire industrial construction project. A special purpose large scale simulation modeling system tailored for industrial construction was designed and developed. It can be used efficiently to build virtual project management laboratories for testing different management strategies. A case model was built and various experiments were executed.

Based on practical application and theoretical analysis, a list of challenges in developing production-based large scale construction simulation was identified. The strategies for overcoming these challenges are recognized as increasing knowledge standardization and reuse, model decomposability, computing ability, product representation, model openness, and model development (or data manipulation) views. These strategies are foundational for increasing the capacity of large scale construction simulation models and for improving the cost-effectiveness of its development.

A literature review was conducted again to look for solutions to the identified challenges, guided by the above recognized strategies. Benchmarks in simulation for both the manufacturing industry and in the military, as well as applications of IT advancements in simulation were investigated. Theories and techniques such as HLA, ontology, IFC, and XML were explored to provide solutions. The prototype integrated architecture was subsequently designed based on the proposed solutions.

The full implementation of the proposed architecture is beyond the scope of this thesis. However, part of the system has been accomplished by the author in collaboration with the NSERC/Alberta Construction Industry Research Chair simulation team. A case study for industrial construction is conducted based on the available developed components of the proposed system.

7.2 SUMMARY OF RESEARCH CONTRIBUTIONS

7.2.1 Academic Contributions

- Development of a special purpose modeling system for modeling industrial construction projects. The industrial construction modeling system (ICMS) is capable of modeling both the product and the information while addressing the quality of their uniqueness. The modeling system is able to model the entire construction stage of industrial construction.
- Development of a list of very useful modeling elements in the *Simphony* environment such as: *DatabaseImporter, DatabaseExporter, TimeCollector, Batch, Unbatch, Assembly, ResourceTracer, CapacityDetector, KanbanSender, and KanbanReceiver*. They are not limited to the simulation of industrial construction.

They can be used with other templates in *Simphony* to enhance their modeling ability. The interoperability was proven in the model development of the phase one research.

- Elucidation of challenges faced in the development of production-based large scale construction simulation using current construction simulation tools; developing the strategies to overcome these identified challenges. These strategies are increasing knowledge standardization and reuse, model decomposability, computing ability, product representation, model openness, and model development (or data manipulation) views.
- Development of solutions to the identified challenges through the exploration and analysis of advanced methodologies and techniques, such as HLA, ontology, IFC, and XML.
- Integration of these solutions into a prototype of development architecture, which promotes the above recognized strategies. The architecture increased the capacity and cost-effectiveness of production-based large scale construction simulation. Part of the system was developed by the author and the NSERC/Alberta Construction Industry Research Chair simulation team.

7.2.2 Contributions to the Construction Industry

- This work is the first to simulate a pipe spool fabrication shop using the simulation-based approach as a tool to facilitate the implementation of lean production techniques in spool fabrication. It proves scientifically that the lean principle of flow can significantly improve the production performance of pipe spool fabrication shops.

- The developed ICMS will allow construction engineers who are not simulation experts to build simulation models efficiently for the entire industrial construction system, including the drafting, material procurement and supply, spool fabrication, module assembly, and site installation stages. These models will be detailed at the production level. A developed simulation model serves construction engineers as a virtual project management laboratory (VPML). In this research, the author built a VPML for an industrial construction contractor using the developed ICMS. It is the first to simulate the drafting process, to model information flow, and to simulate the entire construction portion of industrial construction projects at the production level.
- The VPML will help construction engineers to understand the production performance of industrial construction. It will help construction engineers to test lean production/construction techniques, lean project delivery system strategies, and other construction management theories at a low cost. It will also help test the impacts of such activities as drawing revisions, material delivery strategies, and rework. Contractors can use the tool as a marketing tool to help the client to understand the mechanics of the project and to choose appropriate project delivery strategies. Simulation results can be used for other applications such as scheduling and resource planning.
- Knowledge of industrial construction was acquired, formalized, and standardized. The standardized knowledge was stored in the CSOM library for industrial construction. The knowledge can be reused by future simulation developers.

- Knowledge standardization, accumulation, and reuse will ease the modeling effort along with the modeling practice. This will expedite the application of simulation in the construction industry.

7.3 RECOMMENDATIONS FOR FUTURE RESEARCH

There is not a final and perfect destination at which the strategies of increasing knowledge standardization and reuse, model decomposability, computing ability, product representation, model openness, and model development (or data manipulation) views will have been achieved. The research undertaken in this thesis improved the capacity and cost-effectiveness of production-based large scale construction simulation, but it has yet to realize completely the proposed solutions. Based on the development of phase two research and as suggested in additional literature, several areas of future research can be recommended.

7.3.1 Detail Modeling System of Site Installation

The author developed a set of templates for modeling production systems of industrial construction. The template Site Installation, due to time constraints, was not sufficiently detailed. It is suggested that more complete research be conducted into site construction with a focus on the piping function. Knowledge of division, resource, and process needs to be collected and formalized at a more detailed level. Other modeling elements may be created if necessary.

7.3.2 Enrich the Knowledge Library

Standardization is an iterative task. Three ways to acquire knowledge were proposed in this research. Knowledge required for simulation, however, can be completely defined and standardized only through development practices. The author acquired knowledge of industrial construction during the simulation of industrial construction. The library needs to be realized more completely by future construction simulation developers who are professionals in other domains.

7.3.3 Objectify Standard Processes

In the proposed prototype architecture, the knowledge of process is not included in the CSOM. It is proposed that the knowledge be modeled in federates and stored in database libraries. The process knowledge is a component of some standard data models, such as the IFC model and ontology model, and is objectified. Logics working between processes, relationships between processes and products, information, and resources are included. These models are good sources for developing simulation-related process knowledge. The essential issue, however, is that the executable simulation model is different than data models. The means by which the objectified standard processes can be utilized and the technique for automatically transforming the relationships between products, information, resources, and process into simulation models warrants further research.

The solution to this issue can be a set of rules in the format of translators. The process knowledge can be objectified and stored in the CSOM library. They can be translated into a simulation model by the translator. This advancement will greatly ease simulation model development.

7.3.4 A Standard Data Model in XML Format Simulation Model

In the manufacturing industry, XML is used to develop neutral simulation models. NIST has developed a standard Shop Data Model using XML (Mclean et al. 2002). A shop simulation model can be developed using XML following the Shop Data Model design. It can then be translated by a code generator into an executable simulation model in QUEST. This process was discussed in Section 5.5.2. The updated information regarding the above research is that the data standard is being embedded into the XML format QUEST simulation models. The result will remove the added step of translating an XML neutral model into an executable QUEST model.

In Section 6.6, multi-view model development or data manipulation is proposed and implemented. This implementation, however, is only based on the understanding of the existing model data format and on the querying of the information. If a standard construction simulation data model can be built and presented in XML format, it will be helpful to manipulate data or models. In building such a XML data model, however, it needs to be compatible with the XML simulation model schema, which is governed by the HLA framework and the hierarchical graphic modeling method. The combination of the schemas required by different specifications poses a challenge.

It is recommended that further research of the XML schema of the HLA-compliant simulation model and an optimal schema of XML-based construction simulation models be conducted. This knowledge will significantly improve the functions of the proposed simulation modeling architecture discussed in Section 6.6.

7.4 REFERENCES

Lu, R., Qiao, G., and McLean, C. (2003). "NIST XML Simulation Interface Specification at Boeing: A Case Study." *Proceedings of the 2003 Winter Simulation Conference*, ed. Stephen E. Chick, Paul J. Sánchez, David Ferrin, and Douglas J. Morrice, 1230–1237.

McLean, C., Jones, A., Lee, T., and Riddick, F. (2002). "An Architecture for a Generic Data-Driven Machine Shop Simulator." *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes. 1108–1116.

Qiao, G., Riddick, F., and McLean, C. (2003). "Data Driven Design And Simulation System Based on XML." *Proceedings of the 2003 Winter Simulation Conference*, ed. Stephen E. Chick, Paul J. Sánchez, David Ferrin, and Douglas J. Morrice, 1143–1148.

APPENDIX 1 – INDUSTRIAL CONSTRUCTION SIMULATION MODELING SYSTEM USER'S GUIDE

1. OVERVIEW OF MODELING SYSTEM

The large scale industrial construction modeling system is a special-purpose simulation tool that allows users to develop simulation models for industrial construction projects. This document provides a guide for model development using this modeling system.

The modeling system consists of the Product and Information Modeling and the Production System Modeling, which are described in Chapter 3. Product and information definition data are modeled and stored in the central DBMS. The production system is modeled in *Simphony*. Simulation outputs are also stored in the central DBMS. Microsoft Access is used in this document to implement the database design and implementation.

2. PRODUCT AND INFORMATION MODELING SYSTEM

Within the context of the modeling system, spool, shop drawings, and modules are represented by flow entities. The structure and attributes of these entity models have been described in detail in Chapter 3.

Extracting data contained in CAD drawings or other applications can simplify the process of collecting product/information definition data, such as physical attributes of spools. The data exchange interface between the central DBMS and a CAD system or an

ERP system is dependant upon the characteristics of this system. The interface needs to be designed for a specific system. The design task will not be discussed further.

3. PRODUCTION MODELING SYSTEM

The production modeling system consists of a set of special-purpose oriented simulation templates, which enables the user to model an industrial construction system, and a general-purpose simulation tool. The general-purpose simulation tool used in this system is the Common Template offered in *Simphony*. The following sections describe the function of each element in the special-purpose templates in detail. More information regarding the Common Template can be found in the *Simphony* User Manual.

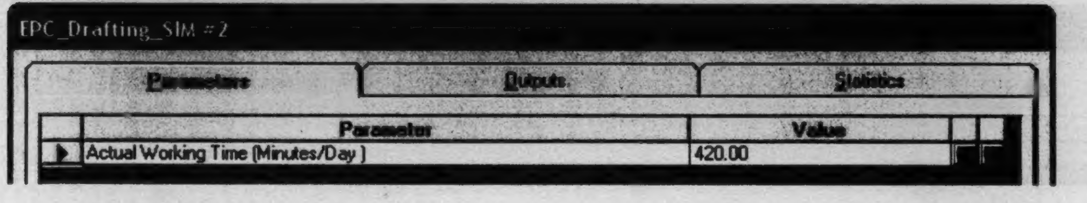
Drafting Template

DraftSIM

The *DraftSIM* element is the parent element of the model for drafting. It represents the whole drafting department, under which drafting processes are modeled.

Input Parameters:

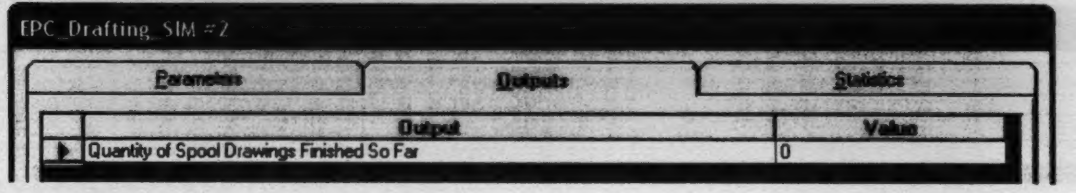
Actual Working Time (Minutes/Day): The actually working time (minutes) in a working day. It can be calculated by deducting lunch time and break time from the total office time in the course of a work day.



Parameter	Value
Actual Working Time (Minutes/Day)	420.00

Output:

Quantity of Spool Drawings Finished So Far: Shows the number of spool drawings finished at the end of the simulation.

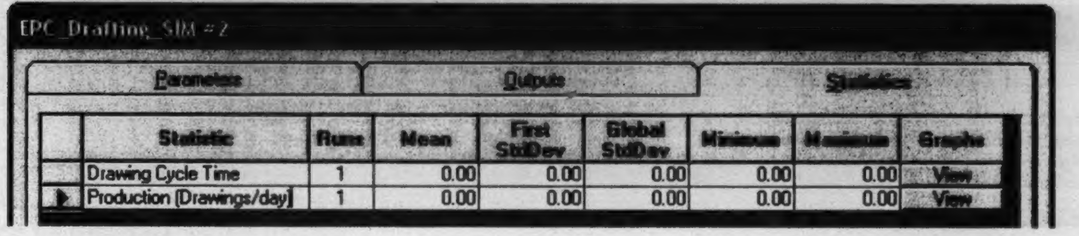

EPC_Drafting_SIM #2
Parameters
Outputs
Statistics

	Output	Value
▶	Quantity of Spool Drawings Finished So Far	0

Statistics:

Drawing Cycle Time: Shows statistics on drawing cycle time during the simulation session (finish time – start time)

Production (Drawings/day): Shows the total number of drawings finished in a working day (number of finished drawings/total days)


EPC_Drafting_SIM #2
Parameters
Outputs
Statistics

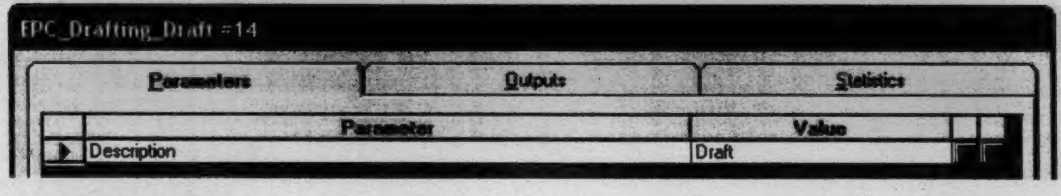
	Statistic	Runs	Mean	First StdDev	Global StdDev	Minimum	Maximum	Graphs
	Drawing Cycle Time	1	0.00	0.00	0.00	0.00	0.00	View
▶	Production (Drawings/day)	1	0.00	0.00	0.00	0.00	0.00	View

Draft

The *Draft* element is a parent element, in which the draftsperson drafts a package of drawings.

Input Parameters:

Description: A text to describe this element in the model.


EPC_Drafting_Draft =14
Parameters
Outputs
Statistics

	Parameter	Value
▶	Description	Draft

Output and Statistics:

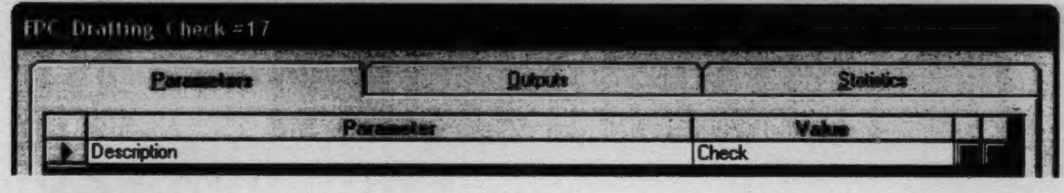
None.

✓ Check

The *Check* element is a parent element used for modeling in which the checker checks a package of drawings.

Input Parameters:

Description: A text to describe this element in the model.


EPC_Drafting_Check =17
Parameters
Outputs
Statistics

	Parameter	Value
▶	Description	Check

Output and Statistics:

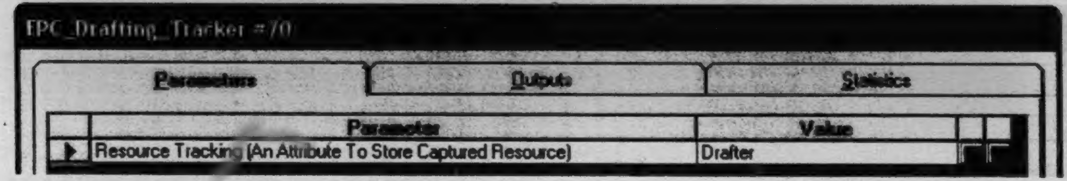
None.

ResourceTracer

The *ResourceTracer* element tracks the draftsperson or checker who drafts or checks the drawing for each drawing package.

Input Parameters:

Resource Tracking (An Attribute to Store Captured Resource): An attribute added to the passing entity (a drawing package) to indicate what type of resource to track.


EPC_Drafting_Tracker #70
Parameters
Outputs
Statistics

Parameter	Value
Resource Tracking (An Attribute To Store Captured Resource)	Drafter

Output and Statistics:

None.

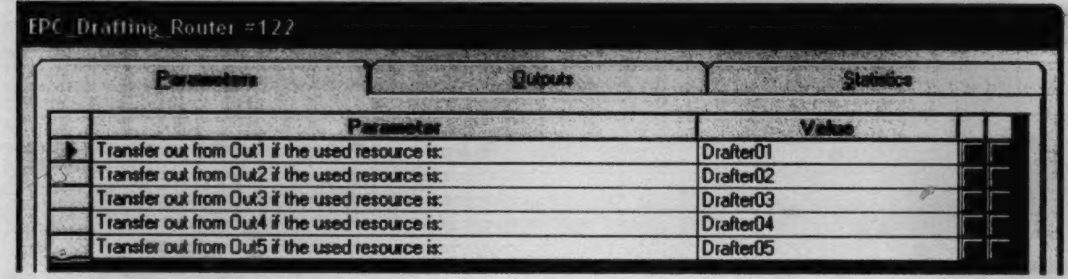
ResourceRouter

The *ResourceRouter* element dispatches the drawing packages back to the draftsperson or checker who drafted or checked them during the process of back drafting or back checking.

Input Parameters:

Transfer out from Out1 if the used resource is: Chooses a resource from the resource list in the dropdown list box, in which all resources in the model are listed. The entity that uses this resource will be transferred out from out port 1.

So does every other input parameter.


EPC_Drafting_Router #122
Parameters
Outputs
Statistics

Parameter	Value
Transfer out from Out1 if the used resource is:	Drafter01
Transfer out from Out2 if the used resource is:	Drafter02
Transfer out from Out3 if the used resource is:	Drafter03
Transfer out from Out4 if the used resource is:	Drafter04
Transfer out from Out5 if the used resource is:	Drafter05

Output and Statistics:

None.

BackDraft

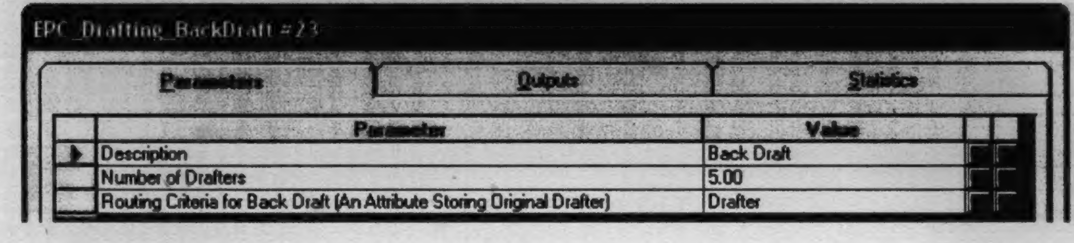
The *BackDraft* element is a parent element, under which drawings are back-drafted by the draftsperson who drafted it during the drafting process.

Input Parameters:

Description: A text to describe this element in the model.

Number of Drafters: Indicates the number of drafters in the model. It will determine the number of outports for the element *ResourceRouter* in its child window.

Routing Criteria for Back Draft (An Attribute Storing Original Drafter): Indicates the attribute of the passing entity (a drawing package). The entity is dispatched under this element according to the value of this attribute.



Parameter	Value
Description	Back Draft
Number of Drafters	5.00
Routing Criteria for Back Draft (An Attribute Storing Original Drafter)	Drafter

Output and Statistics:

None.

✓ BackCheck

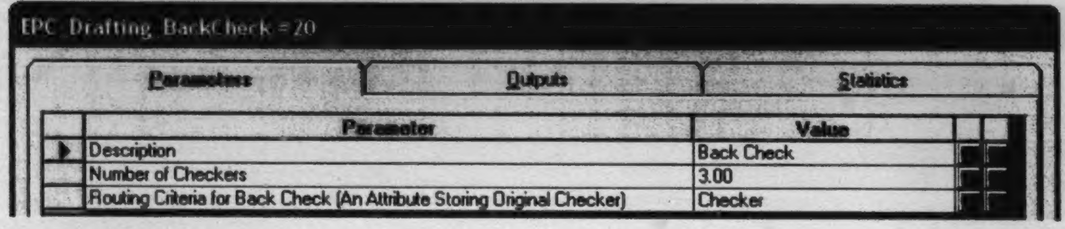
The *BackCheck* element is a parent element, under which drawings are back-checked by the checker who checked it during the checking process.

Input Parameters:

Description: A text to describe this element in the model.

Number of Checkers: Indicates the number of checkers in the model. It will determine the number of outports for the element *ResourceRouter* in its child window.

Routing Criteria for Back-Check (An Attribute Storing Original Checker): Indicates the attribute of the passing entity (a drawing package). The entity is dispatched under this element according to the value of this attribute.


EPC_Drafting_BackCheck =20
Parameters
Outputs
Statistics

Parameter	Value
Description	Back Check
Number of Checkers	3.00
Routing Criteria for Back Check (An Attribute Storing Original Checker)	Checker

Output and Statistics:

None.

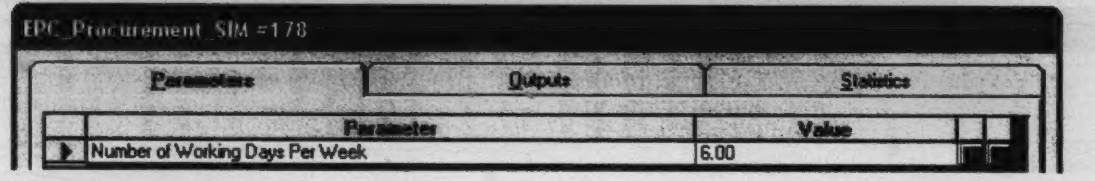
Material Procurement Template

ProcurementSIM

The *ProcurementSIM* element is the parent element of the model of the material procurement and delivery. It represents the whole material procurement process including material delivery by owner.

Input Parameters:

Number of Working Days Per Week: The actually working time (days) in a week.


EPC_Procurement_SIM =178
Parameters
Outputs
Statistics

Parameter	Value
Number of Working Days Per Week	6.00

Output and Statistics:

None.

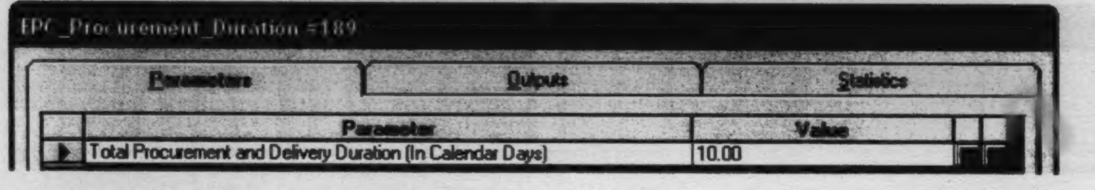
Procurement&Delivery

Represents the whole duration of material procurement and delivery.

Input Parameters:

Total Procurement and Delivery Duration (In Calendar Days): Indicates the duration of material procurement and delivery in calendar days. This duration will be transformed into actual work time (minutes) by the formula:

```
Duration=ob("Duration")/7*ob.Parent("WorkingDays")*7*60
```



Parameter	Value
Total Procurement and Delivery Duration (In Calendar Days)	10.00

Output and Statistics:

None.

Spool Fabrication Template

FabPlant

The *FabPlant* element represents a fabrication plant. It is the parent element of the shop fabrication model.

Input Parameters:

Fabrication Plant Name: A text to describe the fabrication plant in the model.

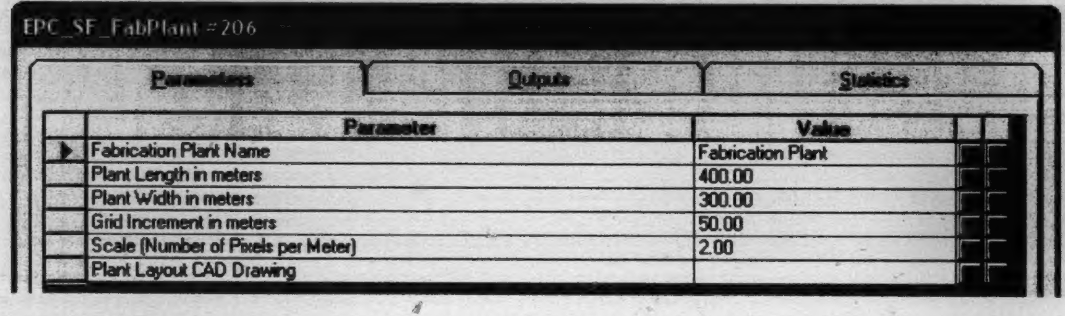
Plant Length in meters: A number to indicate the length of the plant in meters.

Plant Width in meters: A number to indicate the width of the plant in meters.

Grid Increment in meters: A number to indicate the grid increment of length and width.

Scale (Number of Pixels per Meter): A number to indicate the number of pixels per meter. The default value is 2.

Plant Layout CAD Drawing: A file of the CAD drawing of the plant layout. The drawing can be displayed in the child model.



Parameter	Value
Fabrication Plant Name	Fabrication Plant
Plant Length in meters	400.00
Plant Width in meters	300.00
Grid Increment in meters	50.00
Scale (Number of Pixels per Meter)	2.00
Plant Layout CAD Drawing	

Output and Statistics:

None.

Shop

Represents a shop. It can be a fabrication bay, a stress relief shop, or a paint shop.

Input Parameters:

Shop Name: A text to describe the shop.

Shop Length in meters: A number to indicate the length of the shop in meters.

Shop Width in meters: A number to indicate the width of the shop in meters.

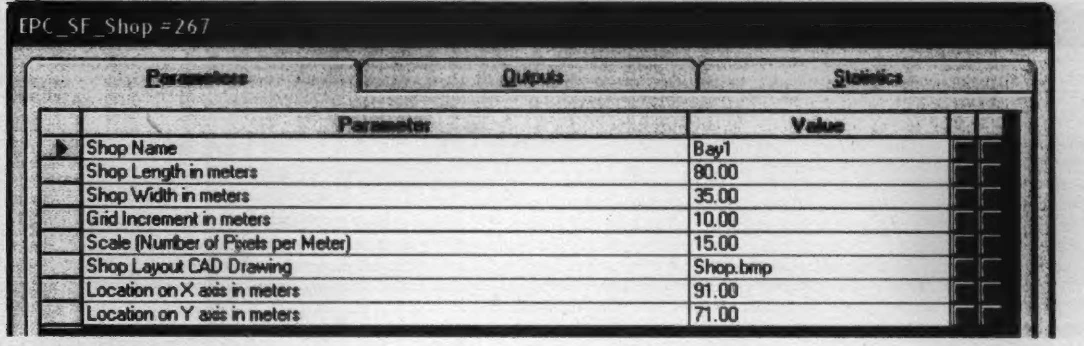
Grid Increment in meters: A number to indicate the grid increment of length and width.

Scale (Number of Pixels per Meter): A number to indicate the number of pixels per meter. The default value is 2.

Shop Layout CAD Drawing: A file of the CAD drawing of the shop layout. The drawing can be displayed in the child model.

Location on X axis in meters: A number to indicate the location of X axis of the shop modeling element.

Location on Y axis in meters: A number to indicate the location of Y axis of the shop modeling element.



Parameter	Value
Shop Name	Bay1
Shop Length in meters	80.00
Shop Width in meters	35.00
Grid Increment in meters	10.00
Scale (Number of Pixels per Meter)	15.00
Shop Layout CAD Drawing	Shop.bmp
Location on X axis in meters	91.00
Location on Y axis in meters	71.00

Output and Statistics:

None.

Workcell

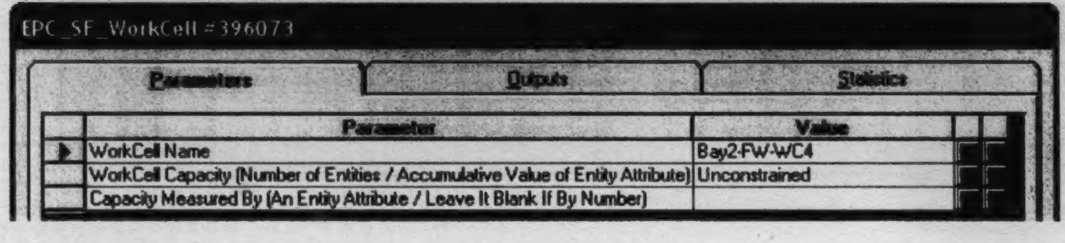
The *workcell* element represents a work cell where a spool is fitted and welded. It has a certain capacity, which only allows user-defined number of spools or diameter inches to be processed. It also can be generalized and used as a working station with certain capacity.

Input Parameters:

WorkCell Name: A text to describe the work cell.

WorkCell Capacity (Number of Entities / Accumulative Value of Entity Attribute): A number indicate the work cell capacity. If it is measured by the total number of entities being processed by the work cell (the value of next attribute is blank), enter the value of the total number. If it is controlled by the attribute value, enter the accumulative value of the entity attribute (e.g. the total diameter inch that a work cell can process at one time).

Capacity Measured By (An Entity Attribute / Leave It Blank If By Number): Indicate the criteria of the work cell capacity. If it is measured by the total number of entities being processed by the work cell, leave this value blank. If it is controlled by the attribute value, such as diameter inch of spools, enter the text of the entity attribute.



Parameter	Value
WorkCell Name	Bay2-FW-WC4
WorkCell Capacity (Number of Entities / Accumulative Value of Entity Attribute)	Unconstrained
Capacity Measured By (An Entity Attribute / Leave It Blank If By Number)	

Output:

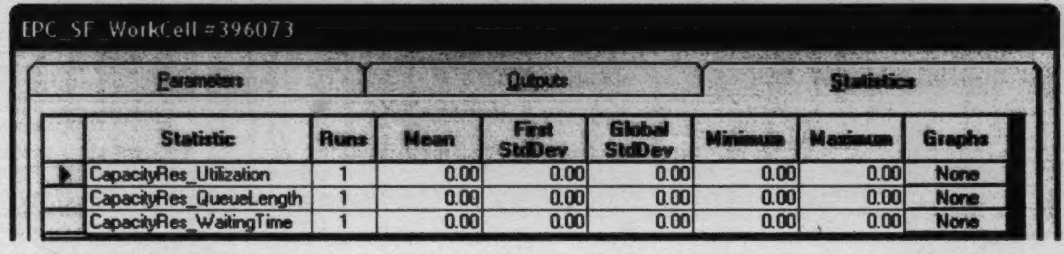
None.

Statistics:

CapacityRes_Utilization: Shows the utilization of the capacity resource of the work cell as a percentage (busytime/total time).

CapacityRes_QueueLength: Shows statistics on queue length of the work cell during the simulation session.

CapacityRes_WaitingTime: Shows statistics on waiting time during the simulation session.



Statistic	Runs	Mean	First StdDev	Global StdDev	Minimum	Maximum	Graphs
CapacityRes_Utilization	1	0.00	0.00	0.00	0.00	0.00	None
CapacityRes_QueueLength	1	0.00	0.00	0.00	0.00	0.00	None
CapacityRes_WaitingTime	1	0.00	0.00	0.00	0.00	0.00	None

LaydownArea

The *LaydownArea* element represents a buffer area where material/products can stay and wait for next process. It has a certain capacity, which only allows user defined-number of entities or user-defined accumulative value of an attribute of entities to stay.

Input Parameters:

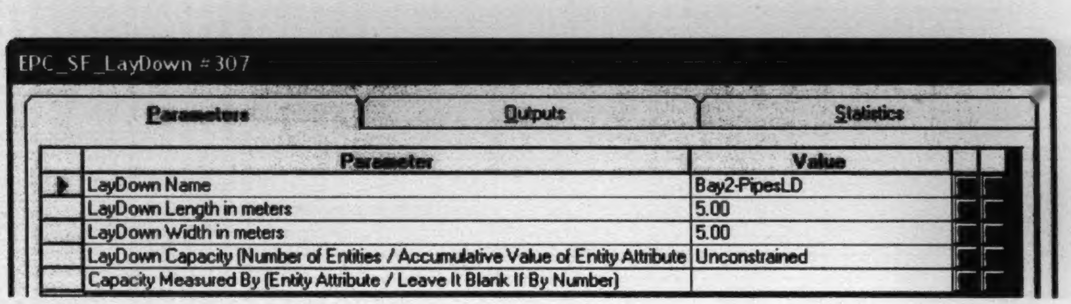
LayDown Name: A text to describe the lay down area.

LayDown Length in meters: A number to indicate the length of the lay down area in meters.

LayDown Width in meters: A number to indicate the length of the lay down area in meters.

LayDown Capacity (Number of Entities / Accumulative Value of Entity Attribute): A number indicate the lay down area capacity. If it is measured by the total number of entities being stored in the lay down area (the value of next attribute is blank), enter the value of the total number. If it is controlled by the attribute value, enter the accumulative value of the entity attribute (e.g. the total diameter inch that a lay down area can hold).

Capacity Measured By (An Entity Attribute / Leave It Blank If By Number): Indicate the criteria of the lay down area capacity. If it is measured by the total number of entities laying down in the lay down area, leave this value blank. If it is controlled by the attribute value, such as diameter inch of spools, enter the text of the entity attribute.



Parameter	Value
LayDown Name	Bay2-PipesLD
LayDown Length in meters	5.00
LayDown Width in meters	5.00
LayDown Capacity (Number of Entities / Accumulative Value of Entity Attribute	Unconstrained
Capacity Measured By (Entity Attribute / Leave It Blank If By Number)	

Output:

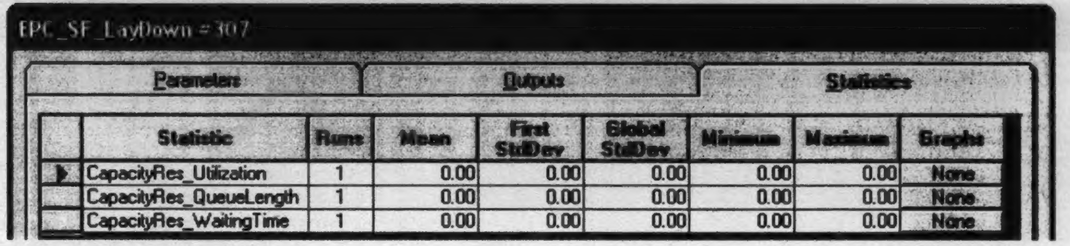
None.

Statistics:

CapacityRes_Utilization: Shows the utilization of the capacity resource of the lay down area as a percentage (busytime/total time).

CapacityRes_QueueLength: Shows statistics on queue length of the lay down area during the simulation session.

CapacityRes_WaitingTime: Shows statistics on waiting time during the simulation session.



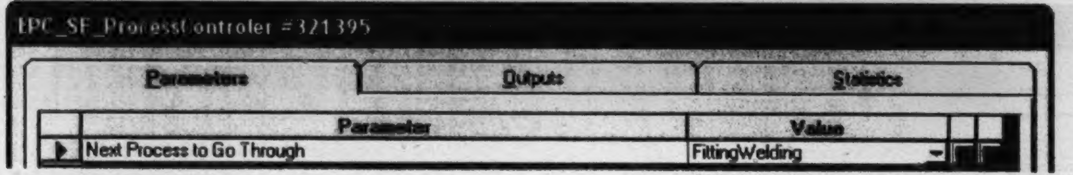
Statistic	Runs	Mean	First StdDev	Global StdDev	Minimum	Maximum	Graphs
CapacityRes_Utilization	1	0.00	0.00	0.00	0.00	0.00	None
CapacityRes_QueueLength	1	0.00	0.00	0.00	0.00	0.00	None
CapacityRes_WaitingTime	1	0.00	0.00	0.00	0.00	0.00	None

ProcessController

The *ProcessController* element simulates a decision maker to decide if the spool goes to next standard process or not.

Input Parameters:

Next Process to Go Through: Choose a process from the dropdown list as the next process to go. It indicates the next standard process from the current process. The information of whether a product goes to that process is carried by the product. A product is sent to different flow when it flows through this element.



Parameter	Value
Next Process to Go Through	FittingWelding

Output and Statistics:

None.

DispatchController

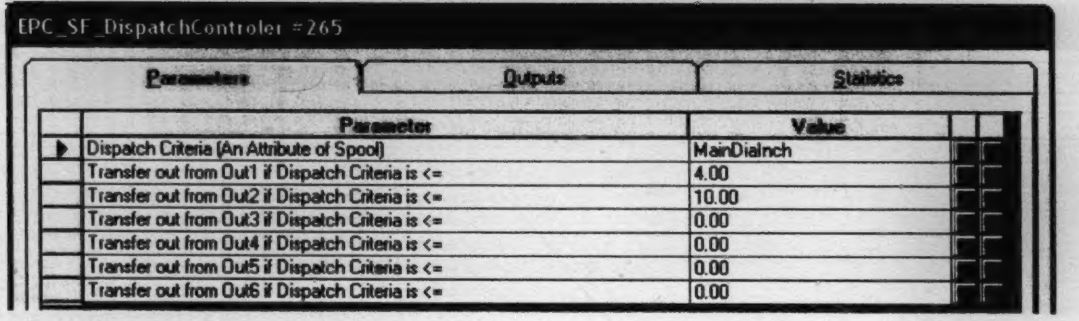
The *DispatchController* element simulates a decision maker to dispatch material/products to one of the destinations that have same process function (e.g. Bays, or WorkCells) based on user-defined criteria.

Input Parameters:

Dispatch Criteria (An Attribute of Spool): An attribute of the passing entity as the criteria for dispatching.

Transfer out from Out1 if Dispatch Criteria is <=: A numeric value to compare with the value of the above attribute. If the attribute value of the passing entity is <= the user-defined value, the passing entity is sent out from the out port 1.

So does each of the remaining parameters.


EPC_SF_DispatchControler =265
Parameters
Outputs
Statistics

Parameter	Value
Dispatch Criteria (An Attribute of Spool)	MainDiaInch
Transfer out from Out1 if Dispatch Criteria is <=	4.00
Transfer out from Out2 if Dispatch Criteria is <=	10.00
Transfer out from Out3 if Dispatch Criteria is <=	0.00
Transfer out from Out4 if Dispatch Criteria is <=	0.00
Transfer out from Out5 if Dispatch Criteria is <=	0.00
Transfer out from Out6 if Dispatch Criteria is <=	0.00

Output and Statistics:

None.

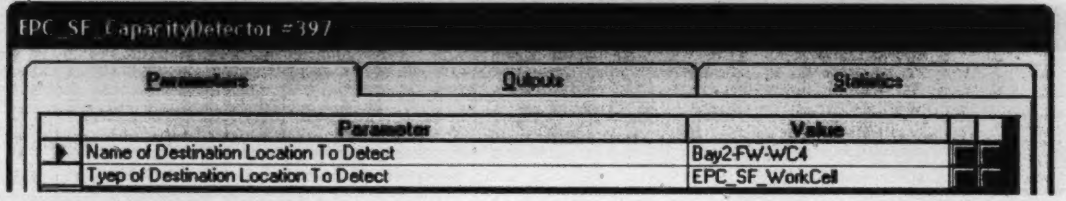
CapacityDetector

The *CapacityDetector* element simulates a decision maker. It can detect the current available capacity of the user-chosen destination (e.g. a WorkCell, or a LayDown.) to determine if a product can go to that destination or not.

Input Parameters:

Name of Destination Location To Detect: Choose a destination location to detect from the dropdown list, which includes all work cells or lay down areas.

Type of Destination Location To Detect: Indicate the type of the destination, work cell or lay down area.


EPC_SF_CapacityDetector =397
Parameters
Outputs
Statistics

Parameter	Value
Name of Destination Location To Detect	Bay2-FW-WC4
Tyep of Destination Location To Detect	EPC_SF_WorkCell

Output and Statistics:

None.

Resource

The *Resource* element represents a container of any equipment or laborer. When a *Resource* element is created, a resource element and a file element of common template are automatically created in the child window. Various interruptions can be modeled in the child window.

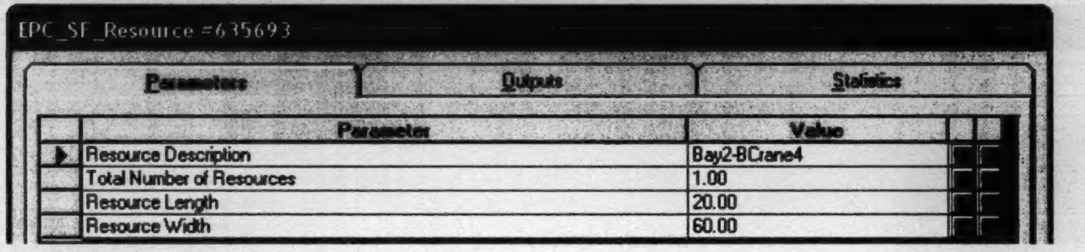
Input Parameters:

Resource Description: A text to describe the resource.

Total Number of Resources: A number to indicate the number of resource.

Resource Length: A number to indicate the length of the resource. It can adjust the size the resource to match the layout of a shop or yard.

Resource Width: A number to indicate the width of the resource. It can adjust the size the resource to match the layout of a shop or yard.



Parameter	Value
Resource Description	Bay2-BCrane4
Total Number of Resources	1.00
Resource Length	20.00
Resource Width	60.00

Output and Statistics:

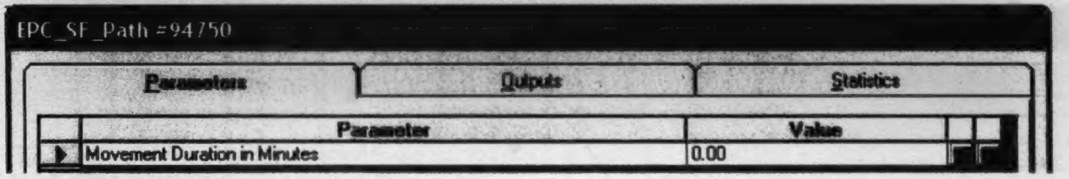
None.

Path

The *Path* element represents a route that products are handled through from a location to another location. The duration of the movement can be defined.

Input Parameters:

Movement Duration in Minutes: A number to indicate the movement duration along the path in minutes.



	Parameter	Value
▶	Movement Duration in Minutes	0.00

Output and Statistics:

None.

DrawingTool (3 implicit elements)

The 3 DrawingTool elements are implicit. They can create layout gridlines and can import plant and shop layout drawings.

Input Parameters:

None.

Output and Statistics:

None.

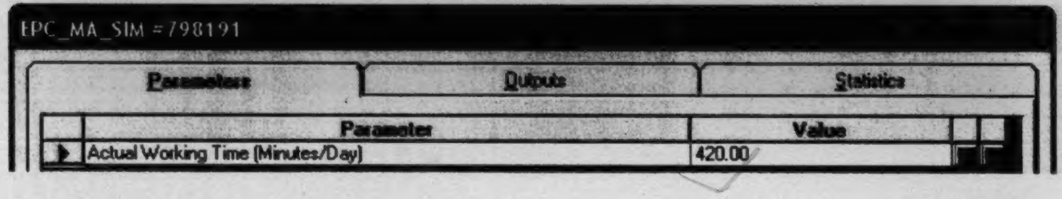
Module Assembly Template

ModuleSIM

The *ModuleSIM* element is the parent element of the model of the module assembly. It represents the whole module assembly yard, under which module assembly processes are modeled.

Input Parameters:

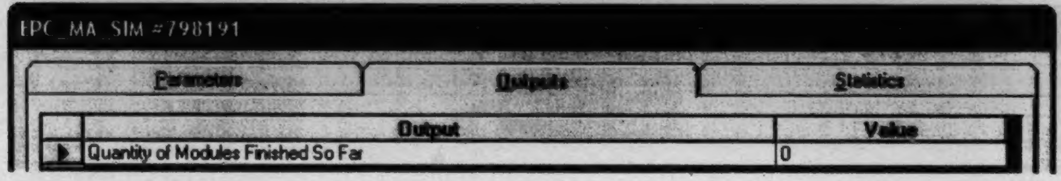
Actual Working Time (Minutes/Day): The actually working time (minutes) in a working day. It can be calculated by deducting lunch time, break time from the total time office time in a working day.


EPC_MA_SIM =798191
Parameters
Outputs
Statistics

Parameter	Value
Actual Working Time (Minutes/Day)	420.00

Output:

Quantity of Modules Finished So Far: Shows the number of modules finished at the end of the simulation.

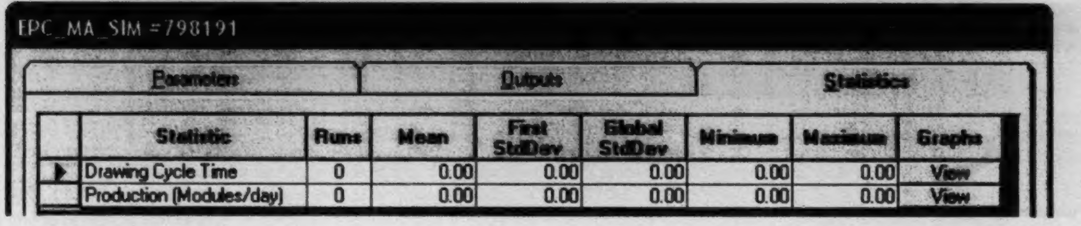

EPC_MA_SIM =798191
Parameters
Outputs
Statistics

Output	Value
Quantity of Modules Finished So Far	0

Statistics:

Module Cycle Time: Shows statistics on module cycle time during the simulation session (finish time – start time)

Production (Modules/day): Shows the total number of modules finished in a working day (number of finished modules/total days)


EPC_MA_SIM =798191
Parameters
Outputs
Statistics

Statistic	Runs	Mean	First StdDev	Global StdDev	Minimum	Maximum	Graphs
Drawing Cycle Time	0	0.00	0.00	0.00	0.00	0.00	View
Production (Modules/day)	0	0.00	0.00	0.00	0.00	0.00	View

|st **1stLayer**

The *1stLayer* element is a parent element for all production activities at the 1st layer of a module.

Input Parameters:

None.

Output and Statistics:

None.

2nd2ndLayer

The *2ndLayer* element is a parent element for all production activities at the 2nd layer of a module.

Input Parameters:

None.

Output and Statistics:

None.

3rd3rdLayer

The *3rdLayer* element is a parent element for all production activities at the 3rd layer of a module.

Input Parameters:

None.

Output and Statistics:

None.

RemainingWork

The *RemainingWork* element is a parent element for all remaining production activities after spool installation at the last layer of a module.

Input Parameters:

None.

Output and Statistics:

None.

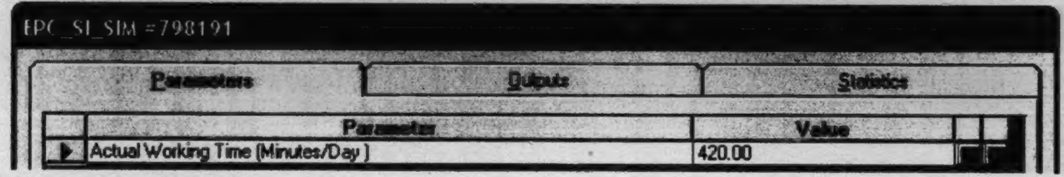
Site Installation Template

SiteSIM

The *SiteSIM* element is the parent element of the model of the site installation. It represents the whole construction site, under which site installation of pipe are modeled.

Input Parameters:

Actual Working Time (Minutes/Day): The actually working time (minutes) in a working day. It can be calculated by deducting lunch time, break time from the total time office time in a working day.



Parameter	Value
Actual Working Time (Minutes/Day)	420.00

Output and Statistics:

None.

Public Module Elements Template

DatabaseImporter

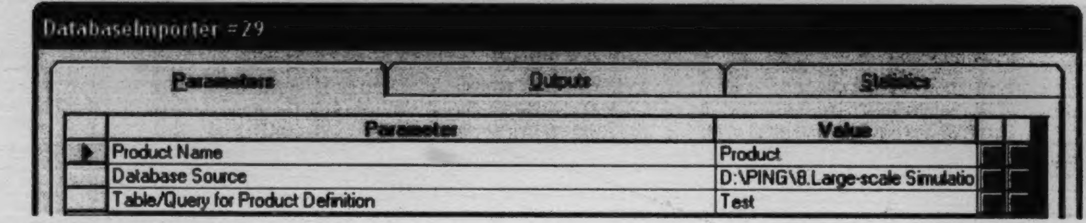
The *DatabaseImporter* element imports products or information (drawings) defined by the Product & Information model in the central database to the Production System model. The path is defined by user in the input parameters.

Input Parameters:

Product Name: A text to describe the product name.

Database Source: A path of database file, which the element will import data from.

Table/Query for Product Definition: The name of the table/query of the data to import in the database file.



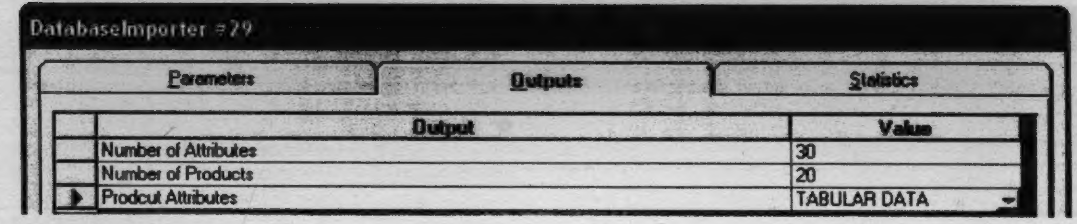
	Parameter	Value
▶	Product Name	Product
	Database Source	D:\PING\8.Large-scale Simulatio
	Table/Query for Product Definition	Test

Output:

Number of Attributes: Shows the total number of attributes of an imported product/drawing.

Number of Products: Shows the total number of imported products/drawings.

Product Attributes: Shows a table of all imported products/drawings with attribute values by clicking the drop down list box.


DatabaseImporter #29
Parameters
Outputs
Statistics

	Output	Value
	Number of Attributes	30
	Number of Products	20
▶	Prodcut Attributes	TABULAR DATA

Statistics:

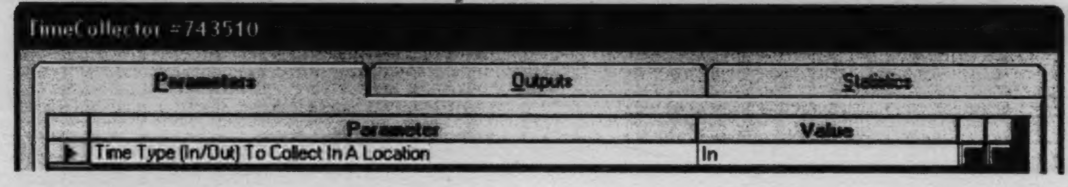
None.

TimeCollector

The *TimeCollector* element can record In time and Out time of an entity at any location (Workcell, Laydown Area, or Shop). They have to be used in a pair in any location to record both In time and Out time. The recorded time are sent to the *DatabaseExporter* element in a model.

Input Parameters:

Time Type (In/Out) To Collect In A Location: Choose a time type to record, In or Out, from the drop down list.


TimeCollector #743510
Parameters
Outputs
Statistics

	Parameter	Value
▶	Time Type (In/Out) To Collect In A Location	In

Output and Statistics:

None.

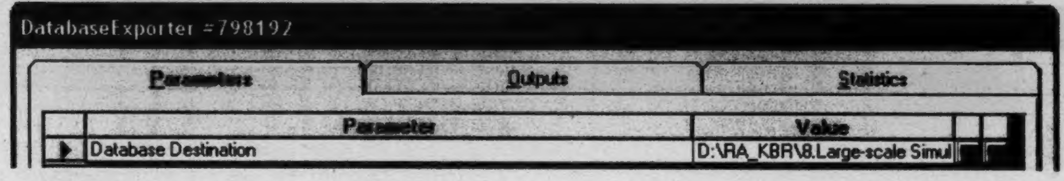
DatabaseExporter

The *DatabaseExporter* element exports all information collected for products from Production System model back to the central database of Product & Information model.

The data are collected through the *TimeCollector* element. They are mainly the In time and Out time of every product passing through a location (Workcell or Laydown Area)

Input Parameters:

Database Destination: A path of database file, which the element will export data to.

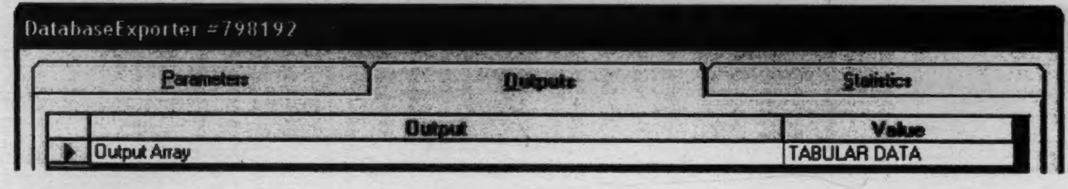


DatabaseExporter =798192

Parameter	Value
Database Destination	D:\RA_KBR\8.Large-scale Simul

Output:

Output Array: Shows a table of all exported data by clicking the drop down list box.



DatabaseExporter =798192

Output	Value
Output Array	TABULAR DATA

Statistics:

None.

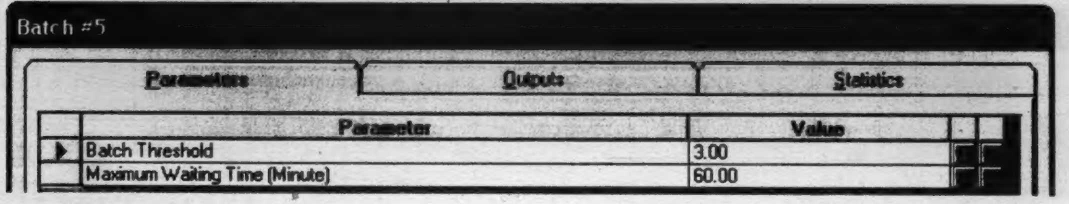
Batch

The *Batch* element can batch a predefined number of entities without changing their attributes. It is always used together with the *Unbatch* element. The batched entities can be released to original state by an Unbatch element without losing any information on each batched entity.

Input Parameters:

Batch Threshold: The number of entities to batch.

Maximum Waiting Time (Minute): Sometime, all left entities to pass this element is less the predefined batch threshold or some entities wait in the element for a long time to accumulate to the batch threshold number. In these cases, the arrived entities can be batched and be released when their waiting time is up to the predefined Maximum Waiting Time even though the number of arrived entities is less than the Batch Threshold.



Parameter	Value
Batch Threshold	3.00
Maximum Waiting Time (Minute)	60.00

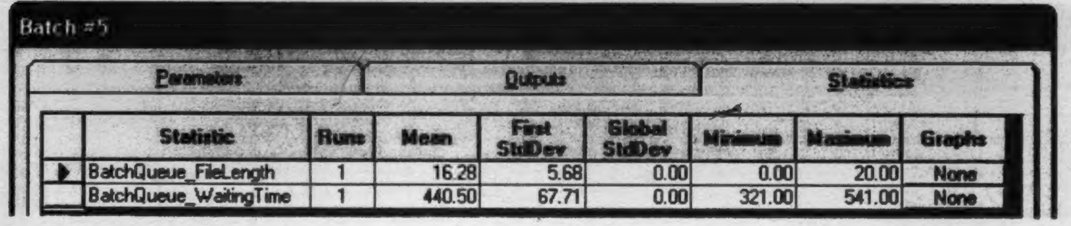
Output:

None.

Statistics:

BatchQueue_FileLength: Shows statistics on queue length of the Batch element during the simulation session.

BatchQueue_WaitingTime: Shows statistics on waiting time during the simulation session.



Statistic	Runs	Mean	First StdDev	Global StdDev	Minimum	Maximum	Graphs
BatchQueue_FileLength	1	16.28	5.68	0.00	0.00	20.00	None
BatchQueue_WaitingTime	1	440.50	67.71	0.00	321.00	541.00	None

Unbatch

The *Unbatch* element can is always used together with the *Batch* element. The batched entities can be released to original state when passing the Unbatch element. Each entity still remains its original attribute value.

Input Parameters:

None.

Output and Statistics:

None.

Assembly

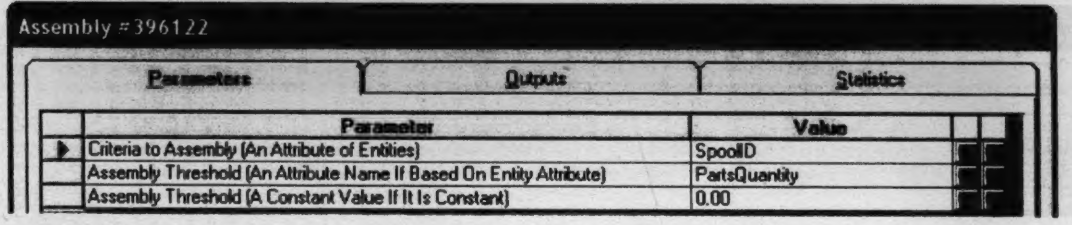
The *Assembly* element is of intelligent. It can scan and draw the entities with same user-defined attribute value (such as SpoolID) from all entities in the queue of this element. Whenever one more new entity joins the queue, the element rescans the queue. When the number of entities with the same attribute value is equal to the user-defined number, they are merged into one entity by this element and this entity is sent out.

Input Parameters:

Criteria to Assembly (An Attribute of Entities): An attribute name of the passing entities. Entities with same value of this attribute are to be assembled.

Assembly Threshold (An Attribute Name If Based On Entity Attribute): In some cases, the number of entities to assemble is dynamic. It is the value of an attribute of the entities.

Assembly Threshold (A Constant Value If It Is Constant): In some cases, the number of entities to assemble is constant. A user can enter the number here.



	Parameter	Value
▶	Criteria to Assembly (An Attribute of Entities)	SpoolID
	Assembly Threshold (An Attribute Name If Based On Entity Attribute)	PartsQuantity
	Assembly Threshold (A Constant Value If It Is Constant)	0.00

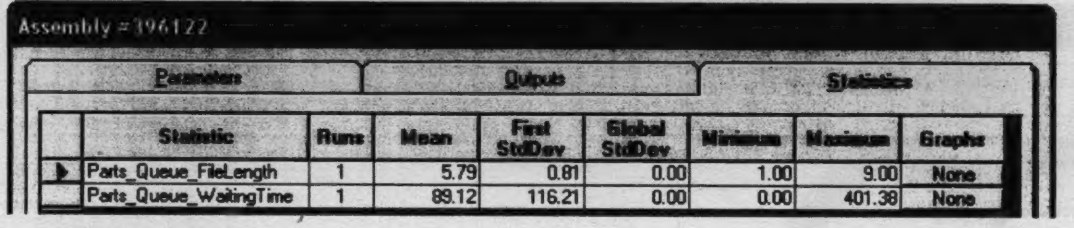
Output:

None.

Statistics:

Parts_Queue_FileLength: Shows statistics on queue length of the during the simulation session.

Parts_Queue_WaitingTime: Shows statistics on waiting time during the simulation session.



	Statistic	Runs	Mean	First StdDev	Global StdDev	Minimum	Maximum	Graphs
▶	Parts_Queue_FileLength	1	5.79	0.81	0.00	1.00	9.00	None
	Parts_Queue_WaitingTime	1	89.12	116.21	0.00	0.00	401.38	None

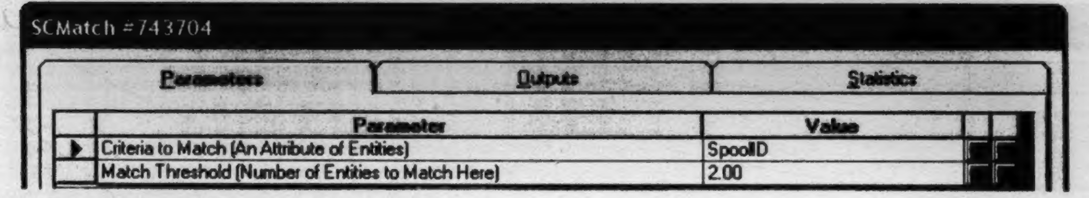
■ SCMatching

The SCMatching element matches and assembles upstream material, pre-fabricated parts, and information for downstream. It works similarly to the *Assembly* element. It is a simplified version of the *Assembly* element.

Input Parameters:

Criteria to Match (An Attribute of Entities): An attribute name of the passing entities. Entities with same value of this attribute are to match.

Match Threshold (Number of Entities to Match Here): The number of entities to match.



Parameter	Value
Criteria to Match (An Attribute of Entities)	SpoolID
Match Threshold (Number of Entities to Match Here)	2.00

Output and Statistics:

None.

KanbanSender

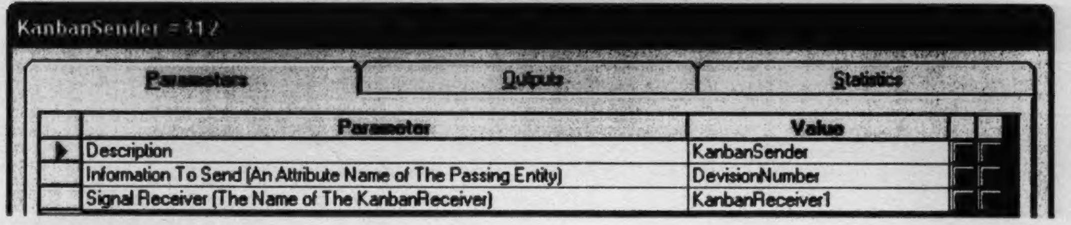
The *KanbanSender* element is always used together with the *KanbanReceiver* element. It sends signals, which are normally values of the passing entities' attribute, from downstream to upstream to pull required materials (drawings, spools, or modules).

Input Parameters:

Description: A text to describe the element.

Information to Send (An Attribute Name of the Passing Entity): An attribute name of the passing entities. It is sent as the signal.

Signal Receiver (The Name of the KanbanReceiver): A text to indicate the *KanbanReceiver* element, which receives the sent signals.



Parameter	Value
Description	KanbanSender
Information To Send (An Attribute Name of The Passing Entity)	DevisionNumber
Signal Receiver (The Name of The KanbanReceiver)	KanbanReceiver1

Output and Statistics:

None.

KanbanReceiver

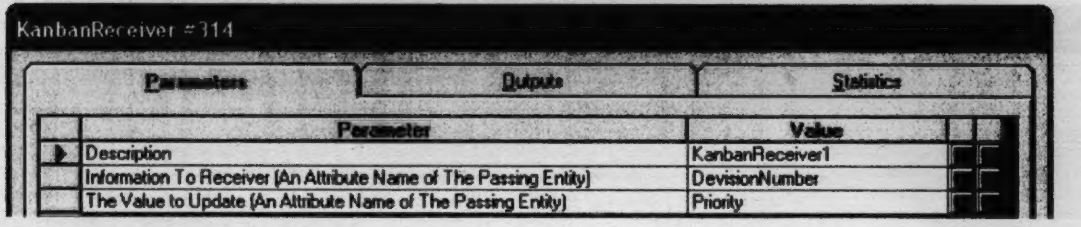
The *KanbanReceiver* element is always used together with the *KanbanSender* element. It receives the signals sent by a *KanbanSender* element and makes response by adjusting the priority of passing entities at upstream.

Input Parameters:

Description: A text to describe the element.

Information To Receiver (An Attribute Name of The Passing Entity): An attribute name of the passing entities. It is the received signal.

The Value to Update (An Attribute Name of The Passing Entity): An attribute name of the passing entities. This value is updated when the entity passes the *KanbanReceiver* element. Normally, it is defined as “Priotity”.



Parameter	Value
Description	KanbanReceiver1
Information To Receiver (An Attribute Name of The Passing Entity)	DevisionNumber
The Value to Update (An Attribute Name of The Passing Entity)	Priority

Output and Statistics:

None.

APPENDIX 2 – DEVELOPMENT CODE OF SIMULATION MODELING SYSTEM FOR INDUSTRIAL CONSTRUCTION

1. DRAFTING TEMPLATE

EPC_Drafting_SIM

```
Public Function EPC_Drafting_SIM_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y
As Single) As Boolean
        EPC_Drafting_SIM_OnCreate=True
        ob.OnCreate x,y,True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+200
        ob.CoordinatesY(1)=y+200

        ob.AddAttribute "ActualWorkingTime", "Actual Working Time (Minutes/Day )", CFC_Numeric,
CFC_Single, CFC_ReadWrite,1
        ob.AddAttribute "FinishedDrawings", "Quantity of Spool Drawings Finished So Far",
CFC_Numeric, CFC_Single, CFC_ReadOnly
        ob.AddStatistic "DrawingCycleTime","Drawing Cycle Time",False,True
        ob.AddStatistic "Production","Production (Drawings/day)",False,True

        ob("ActualWorkingTime")=420

End Function

Public Sub EPC_Drafting_SIM_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.RenderPicture "DraftSIM.jpg", ob.CoordinatesX(0), ob.CoordinatesY(0),
ob.CoordinatesX(1)-ob.CoordinatesX(0), ob.CoordinatesY(1)-ob.CoordinatesY(0)

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_SOLID,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2, ob.CoordinatesY(0)-2, ob.CoordinatesX(1)+2,
ob.CoordinatesY(1)+2
        End If
        ob.DrawConnectionPoints

End Sub

Public Sub EPC_Drafting_SIM_OnSimulationInitializeRun(ob As CFCSim_ModelingElementInstance,
RunNum As Integer)
        ob("FinishedDrawings")=0
End Sub
```

EPC_Drafting_Draft

```
Public Function EPC_Drafting_Draft_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y
As Single) As Boolean
        EPC_Drafting_Draft_OnCreate=True
        ob.OnCreate x,y,True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+100
        ob.CoordinatesY(1)=y+50

        ob.AddAttribute "Description","Description",CFC_Text,CFC_Single,CFC_ReadWrite
        ob("Description")="Draft"
        ob.AddStatistic "DrawingCycleTime","Drawing Cycle Time",False,True
        ob.AddStatistic "Production","Production (SpoolDrawings/day)",False,True

End Function

Public Sub EPC_Drafting_Draft_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub EPC_Drafting_Draft_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)

        CDC.ChangeFont "Arial",14,True,False,False,False
        CDC.TextOut ob.CoordinatesX(0)+20,ob.CoordinatesY(0)+15, ob("Description")

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-2, ob.CoordinatesX(1)+2,
ob.CoordinatesY(1)+2
        End If
        ob.DrawConnectionPoints
```

EPC_Drafting_Check

```
Public Function EPC_Drafting_Check_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y
As Single) As Boolean
        EPC_Drafting_Check_OnCreate=True
        ob.OnCreate x,y,True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+100
        ob.CoordinatesY(1)=y+50

        ob.AddAttribute "Description","Description",CFC_Text,CFC_Single,CFC_ReadWrite
        ob("Description")="Check"
        ob.AddStatistic "DrawingCycleTime","Drawing Cycle Time",False,True
        ob.AddStatistic "Production","Production (SpoolDrawings/day)",False,True
```

```
End Function
Public Sub EPC_Drafting_Check_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub EPC_Drafting_Check_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.Rectangle ob.CoordinatesX(0), ob.CoordinatesY(0), ob.CoordinatesX(1),
ob.CoordinatesY(1)
        CDC.ChangeFont "Arial",14,True,False,False,False
        CDC.TextOut ob.CoordinatesX(0)+20,ob.CoordinatesY(0)+15, ob("Description")

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-2, ob.CoordinatesX(1)+2,
ob.CoordinatesY(1)+2
        End If
        ob.DrawConnectionPoints

End Sub
```

EPC_Drafting_Tracker

```
Public Function EPC_Drafting_Tracker_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single,
y As Single) As Boolean
        ob.OnCreate x,y,True
        EPC_Drafting_Tracker_OnCreate=True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+90
        ob.CoordinatesY(1)=y+50

        ob.AddAttribute "ResourceTracking","Resource Tracking (An Attribute To Store Captured
Resource)",CFC_Text, CFC_Single, CFC_ReadWrite
        ob("ResourceTracking")="Drafter"

        ob.AddConnectionPoint "In" , x-10, y+25, CInput, 5
        ob.AddConnectionPoint "Out",x+100,y+25,COutput,5
End Function

Public Sub EPC_Drafting_Tracker_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub EPC_Drafting_Tracker_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.ChangeFont "Courier New",13,True,False,False,False

        CDC.Rectangle ob.CoordinatesX(0), ob.CoordinatesY(0), ob.CoordinatesX(1),
ob.CoordinatesY(1)

        CDC.MoveTo ob.CoordinatesX(0)+5, ob.CoordinatesY(0)+10
        CDC.LineTo ob.CoordinatesX(0)+30, ob.CoordinatesY(0)+10
        CDC.MoveTo ob.CoordinatesX(0)+5, ob.CoordinatesY(0)+20
        CDC.LineTo ob.CoordinatesX(0)+30, ob.CoordinatesY(0)+20
        CDC.MoveTo ob.CoordinatesX(0)+5, ob.CoordinatesY(0)+30
        CDC.LineTo ob.CoordinatesX(0)+30, ob.CoordinatesY(0)+30
        CDC.MoveTo ob.CoordinatesX(0)+5, ob.CoordinatesY(0)+40
        CDC.LineTo ob.CoordinatesX(0)+30, ob.CoordinatesY(0)+40


        CDC.TextOut ob.CoordinatesX(0)+35,ob.CoordinatesY(0)+15, "Resource"
        CDC.TextOut ob.CoordinatesX(0)+35,ob.CoordinatesY(0)+25, "Tracking"

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)

        ob.DrawConnectionPoints
End Sub

Public Sub EPC_Drafting_Tracker_OnSimulationTransferIn(ob As CFCSim_ModelingElementInstance,
Entity As CFCSim_Entity, SrcCp As CFCSim_ConnectionPoint, DstCp As CFCSim_ConnectionPoint)
   Entity(ob("ResourceTracking"))=Entity ("CEM_Common_RqstdRes")("ResName")
        ob.TransferOut Entity

End Sub
```

EPC_Drafting_Router

```
Public Function EPC_Drafting_Router_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single,
y As Single) As Boolean
        ob.OnCreate x,y,True
        EPC_Drafting_Router_OnCreate=True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+50
        ob.CoordinatesY(1)=y+120

        ob.AddConnectionPoint "In" , x-5, y+60, CInput, 5

        Dim i As Integer
        For i=1 To ob.Parent("NumRes")
                ob.AddAttribute "Routing" & i,"Transfer out from Out" & i & " if the used resource
is:",CFC_Text, CFC_ListBox, CFC_ReadWrite
                ob.AddConnectionPoint "Out" & i,x+55,y+12*i,COutput,5
        Next

End Function

Public Sub EPC_Drafting_Router_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub
```

```
Public Sub EPC_Drafting_Router_OnDraw(ob As CFCSim_ModelingElementInstance)
    CDC.ChangeFont "Courier New",13,True,False,False,False

    CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)

    Dim i As Integer
    For i=1 To ob.Parent("NumRes")
        CDC.TextOut ob.CoordinatesX(0)+25,ob.CoordinatesY(0)-6+12*i, "Out" & i
    Next

    If ob.Selected Then
        CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
        CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
    End If
    CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)

    ob.DrawConnectionPoints
End Sub

Public Sub EPC_Drafting_Router_OnListBoxInitialize(ob As CFCSim_ModelingElementInstance, attr As
CFCSim_Attribute, lstList As Object)
    Dim childob As CFCSim_ModelingElementInstance
    For Each childob In Elements
        If childob.ElementType="Resource" Then
            lstList.addItem childob("ResName")
        End If
    Next

End Sub

Public Sub EPC_Drafting_Router_OnSimulationTransferIn(ob As CFCSim_ModelingElementInstance,
Entity As CFCSim_Entity, SrcCp As CFCSim_ConnectionPoint, DstCp As CFCSim_ConnectionPoint)
    Dim i As Integer

    For i=1 To ob.Parent("NumRes")
        If Entity(ob.Parent("RoutingCriteria"))=ob("Routing" & i) Then
            ob.TransferOut Entity, ob.ConnectionPoints("Out" & i)
            Exit Sub
        End If
    Next

End Sub
```

EPC_Drafting_BackDraft

```
Public Function EPC_Drafting_BackDraft_OnCreate(ob As CFCSim_ModelingElementInstance, x As
Single, y As Single) As Boolean
    EPC_Drafting_BackDraft_OnCreate=True
    ob.OnCreate x,y,True

    ob.SetNumCoordinates 2
    ob.CoordinatesX(0)[illegible]x
```

```
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+100
        ob.CoordinatesY(1)=y+50

        ob.AddAttribute "Description","Description",CFC_Text,CFC_Single,CFC_ReadWrite
        ob("Description")="Back Draft"
        ob.AddAttribute "NumRes","Number of Drafters",CFC_Numeric, CFC_Single, CFC_ReadWrite
        ob.AddAttribute "RoutingCriteria","Routing Criteria for Back Draft (An Attribute Storing
Original Drafter)",CFC_Text, CFC_Single, CFC_ReadWrite
        ob.AddStatistic "DrawingCycleTime","Drawing Cycle Time",False,True
        ob.AddStatistic "Production","Production (SpoolDrawings/day)",False,True

        ob("NumRes")=5
        ob("RoutingCriteria")="Drafter"

End Function

Public Sub EPC_Drafting_BackDraft_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub EPC_Drafting_BackDraft_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.Rectangle ob.CoordinatesX(0), ob.CoordinatesY(0), ob.CoordinatesX(1),
ob.CoordinatesY(1)
        CDC.ChangeFont "Arial",14,True,False,False,False
        CDC.TextOut ob.CoordinatesX(0)+20,ob.CoordinatesY(0)+15, ob("Description")

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        ob.DrawConnectionPoints

End Sub
```

EPC_Drafting_BackCheck

```
Public Function EPC_Drafting_BackCheck_OnCreate(ob As CFCSim_ModelingElementInstance, x As
Single, y As Single) As Boolean
        EPC_Drafting_BackCheck_OnCreate=True
        ob.OnCreate x,y,True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+100
        ob.CoordinatesY(1)=y+50

        ob.AddAttribute "Description","Description",CFC_Text,CFC_Single,CFC_ReadWrite
        ob("Description")="Back Check"
        ob.AddAttribute "NumRes","Number of Checkers",CFC_Numeric, CFC_Single, CFC_ReadWrite
        ob.AddAttribute "RoutingCriteria","Routing Criteria for Back Check (An Attribute Storing
Original Checker)",CFC_Text, CFC_Single, CFC_ReadWrite
```

```
        ob.AddStatistic "DrawingCycleTime","BackChecking Cycle Time",False,True
        ob.AddStatistic "Production","Production (SpoolDrawings/day)",False,True

        ob("NumRes")=5
        ob("RoutingCriteria")="Checker"

End Function

Public Sub EPC_Drafting_BackCheck_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub EPC_Drafting_BackCheck_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.Rectangle ob.CoordinatesX(0), ob.CoordinatesY(0), ob.CoordinatesX(1),
ob.CoordinatesY(1)
        CDC.ChangeFont "Arial",14,True,False,False,False
        CDC.TextOut ob.CoordinatesX(0)+20,ob.CoordinatesY(0)+15, ob("Description")

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        ob.DrawConnectionPoints

End Sub
```

2. MATERIAL PROCUREMENT TEMPLATE

EPC_Procurement_SIM

```
Public Function EPC_Procurement_SIM_OnCreate(ob As CFCSim_ModelingElementInstance, x As
Single, y As Single) As Boolean
        EPC_Procurement_SIM_OnCreate=True
        ob.OnCreate x,y,True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+150
        ob.CoordinatesY(1)=y+100

        ob.AddAttribute "WorkingDays","Number of Working Days Per Week",CFC_Numeric,
CFC_Single, CFC_ReadWrite,0
        ob("WorkingDays")=5
End Function

Public Sub EPC_Procurement_SIM_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.RenderPicture
"MaterialProcurement.jpg",ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1)-
ob.CoordinatesX(0),ob.CoordinatesY(1)-ob.CoordinatesY(0)

        If ob.Selected Then
```

```
                CDC.ChangeLineStyle CFC_SOLID,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        ob.DrawConnectionPoints

End Sub
```

EPC_Procurement_Duration

```
Public Function EPC_Procurement_Duration_OnCreate(ob As CFCSim_ModelingElementInstance, x As
Single, y As Single) As Boolean
        ob.OnCreate x,y,True
        EPC_Procurement_Duration_OnCreate=True
        ob.AddAttribute "Duration","Total Procurement and Delivery Duration (In Calendar
Days)",CFC_Numeric, CFC_Single, CFC_ReadWrite,0
        ob("Duration")=20

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+100
        ob.CoordinatesY(1)=y+50

        ob.AddConnectionPoint "In" , x-10, y+25, CInput, 5
        ob.AddConnectionPoint "Out",x+115,y+25,COutput,5
End Function

Public Sub EPC_Procurement_Duration_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub EPC_Procurement_Duration_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.ChangeFont "Courier New",13,True,False,False,False

        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)

        CDC.ChangeFont "Arial",11,True,False,False,False
        CDC.TextOut ob.CoordinatesX(0)+5,ob.CoordinatesY(0)+15, "Delivery Duration"

        If ob("Duration").Calculation=CFC_Simple Then
                CDC.TextOut ob.CoordinatesX(0)+5,ob.CoordinatesY(0)+35, "Dur: " & ob("Duration")
        Else
                CDC.TextOut ob.CoordinatesX(0)+5,ob.CoordinatesY(0)+35, "Dur: " & "(Formula)"
        End If

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)

        ob.DrawConnectionPoints
```

```
End Sub

Public Sub EPC_Procurement_Duration_OnSimulationInitialize(ob As
CFCSim_ModelingElementInstance)
        ob.AddEvent "Start",True
        ob.AddEvent "Finish"
End Sub

Public Sub EPC_Procurement_Duration_OnSimulationProcessEvent(ob As
CFCSim_ModelingElementInstance, MyEvent As String, Entity As CFCSim_Entity)
        Dim Duration As Double

        Select Case MyEvent
                Case "Start"
                                Duration=ob("Duration")/7*ob.Parent("WorkingDays")*7*60
                                ob.ScheduleEvent Entity,"Finish",Duration

                Case "Finish"
                        ob.TransferOut Entity
        End Select
End Sub
```

3. SPOOL FABRICATION TEMPLATE

Function EPC_SF_FabPlant

```
Public Function EPC_SF_FabPlant_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y
As Single) As Boolean
        EPC_SF_FabPlant_OnCreate=True
        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+200
        ob.CoordinatesY(1)=y+150

        ob.AddAttribute "Plant","Fabrication Plant Name",CFC_Text, CFC_Single, CFC_ReadWrite
        ob.AddAttribute "Length","Plant Length in meters",CFC_Numeric, CFC_Single,
CFC_ReadWrite,50,5000
        ob.AddAttribute "Width","Plant Width in meters",CFC_Numeric, CFC_Single,
CFC_ReadWrite,50,5000
        ob.AddAttribute "GridIncrement","Grid Increment in meters",CFC_Numeric, CFC_Single,
CFC_ReadWrite,1,500
        ob.AddAttribute "Scale","Scale (Number of Pixels per
Meter)",CFC_Numeric,CFC_Single,CFC_ReadWrite,1,10
        ob.AddAttribute "DWG","Plant Layout CAD Drawing",CFC_Text, CFC_Single, CFC_ReadWrite

        ob!Plant="Fabrication Plant"
        ob!GridIncrement=50
        ob!length=400
        ob!width=300
        ob!Scale=2
        ob!DWG=""
```

```
    EPC_SF_FabPlant_DrawGrid ob,300,200,50,2

End Function

Public Sub EPC_SF_FabPlant_DrawGrid(ob As CFCSim_ModelingElementInstance, length As
Integer,width As Integer,GridIncrement As Integer,Scale As Integer)
    Dim NewElement As CFCSim_ModelingElementInstance
    Dim i As Integer

    ' delete All child Elements And labels
    For Each NewElement In ob.ChildElements
        If NewElement.ElementType="EPC_SF_Outline" Or NewElement.ElementType =
"EPC_SF_Label" Then
            NewElement.Delete
        End If
    Next

    Set NewElement=ob.AddElement("EPC_SF_Label",CSng(15),CSng(15))
    NewElement!Value=ob!Plant
    ' draw horizontal grid

    For i=0 To width Step GridIncrement
        Set NewElement=ob.AddElement("EPC_SF_Outline",1,1)
        NewElement.CoordinatesX(0)=0
        NewElement.CoordinatesY(0)=i*Scale
        NewElement.CoordinatesX(1)=length*Scale
        NewElement.CoordinatesY(1)=i*Scale
        If i>0 Then
            Set
NewElement=ob.AddElement("EPC_SF_Label",CSng(Scale*length+5),CSng(Scale*i-8))
            NewElement!Value=i & " m"
        End If
    Next

    ' draw vertical grid
    For i=0 To length Step GridIncrement
        Set NewElement=ob.AddElement("EPC_SF_Outline",1,1)
        NewElement.CoordinatesX(0)=i*Scale
        NewElement.CoordinatesY(0)=0
        NewElement.CoordinatesX(1)=i*Scale
        NewElement.CoordinatesY(1)=width*Scale
        If i>0 Then
            Set NewElement=ob.AddElement("EPC_SF_Label",CSng(Scale*i-
15),CSng(Scale*width+5))
            NewElement!Value=i & " m"
        End If
    Next
End Sub

Public Sub EPC_SF_FabPlant_DrawLayout(ob As CFCSim_ModelingElementInstance, length As
Integer,width As Integer,GridIncrement As Integer,Scale As Integer)
    Dim NewElement As CFCSim_ModelingElementInstance
    Dim i As Integer

    ' delete All child outline Elements
    For Each NewElement In ob.ChildElements
```

```
                If NewElement.ElementType="EPC_SF_Layout" Then
                        NewElement.Delete
                End If
        Next

        If Not IsNull(ob("DWG")) And Not IsEmpty(ob("DWG")) Then
                Set NewElement=ob.AddElement("EPC_SF_Layout",1,1)
                NewElement.CoordinatesX(0)=0
                NewElement.CoordinatesY(0)=0
                NewElement.CoordinatesX(1)=length*Scale
                NewElement.CoordinatesY(1)=width*Scale
        End If

End Sub

Public Sub EPC_SF_FabPlant_OnDraw(ob As CFCSim_ModelingElementInstance)
        If ob("DWG")<>"" Then
                EPC_SF_FabPlant_DrawLayout ob,ob!length,ob!width,ob!GridIncrement,ob!Scale
        End If

        CDC.RenderPicture
"Fabrication.jpg",ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1)-
ob.CoordinatesX(0),ob.CoordinatesY(1)-ob.CoordinatesY(0)
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)
        CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_SOLID,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2

        End If
        ob.DrawConnectionPoints
End Sub

Public Function EPC_SF_FabPlant_OnValidateParameters(ob As CFCSim_ModelingElementInstance,
Parameters As Object) As Boolean
        EPC_SF_FabPlant_OnValidateParameters=ob.OnValidateParameters(Parameters,True)
        If Not EPC_SF_FabPlant_OnValidateParameters Then Exit Function
        EPC_SF_FabPlant_DrawGrid
ob,Parameters("Length"),Parameters("Width"),Parameters("GridIncrement"),Parameters("Scale")

        ob.RefreshDisplay
End Function
```

EPC_SF_Shop

```
Public Function EPC_SF_Shop_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y As
Single) As Boolean
        EPC_SF_Shop_OnCreate=True

        ob.AddAttribute "Shop","Shop Name",CFC_Text, CFC_Single, CFC_ReadWrite
        ob.AddAttribute "Length","Shop Length in meters",CFC_Numeric, CFC_Single,
CFC_ReadWrite,0,5000
```

```
        ob.AddAttribute "Width","Shop Width in meters",CFC_Numeric, CFC_Single,
CFC_ReadWrite,1,5000
        ob.AddAttribute "GridIncrement","Grid Increment in meters",CFC_Numeric, CFC_Single,
CFC_ReadWrite,1,500
        ob.AddAttribute "Scale","Scale (Number of Pixels per
Meter)",CFC_Numeric,CFC_Single,CFC_ReadWrite,1,20
        ob.AddAttribute "DWG","Shop Layout CAD Drawing",CFC_Text, CFC_Single, CFC_ReadWrite

        ob.AddAttribute "X","Location on X axis in meters",CFC_Numeric, CFC_Single,
CFC_ReadWrite,0,5000
        ob.AddAttribute "Y","Location on Y axis in meters",CFC_Numeric, CFC_Single,
CFC_ReadWrite,0,5000
        ob!Shop="Fabrication Shop"
        ob!GridIncrement=10
        ob!Length=150
        ob!width=30
        ob!Scale=15
        ob!DWG="Shop.bmp"

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob("X")=x/ob.Parent("Scale")
        ob("Y")=y/ob.Parent("Scale")
        ob.CoordinatesX(1)=x+ob!Length*ob.Parent("Scale")
        ob.CoordinatesY(1)=y+ob!width*ob.Parent("Scale")


Public Sub EPC_SF_Shop_DrawGrid(ob As CFCSim_ModelingElementInstance, Length As Integer,width
As Integer,GridIncrement As Integer,Scale As Integer)
        Dim NewElement As CFCSim_ModelingElementInstance
        Dim i As Integer

        For Each NewElement In ob.ChildElements
                If NewElement.ElementType="EPC_SF_Outline" Or NewElement.ElementType =
"EPC_SF_Label" Then
                        NewElement.Delete
                End If
        Next

        Set NewElement=ob.AddElement("EPC_SF_Label",CSng(15),CSng(15))
        NewElement!Value=ob!Shop

        For i=0 To width Step GridIncrement
                Set NewElement=ob.AddElement("EPC_SF_Outline",1,1)
                NewElement.CoordinatesX(0)=0
                NewElement.CoordinatesY(0)=i*Scale
                NewElement.CoordinatesX(1)=Length*Scale
                NewElement.CoordinatesY(1)=i*Scale
                If i>0 Then
                        Set
NewElement=ob.AddElement("EPC_SF_Label",CSng(Scale*Length+5),CSng(Scale*i-8))
                        NewElement!Value=i & " m"
                End If
        Next
```

```
    For i=0 To Length Step GridIncrement
        Set NewElement=ob.AddElement("EPC_SF_Outline",1,1)
        NewElement.CoordinatesX(0)=i*Scale
        NewElement.CoordinatesY(0)=0
        NewElement.CoordinatesX(1)=i*Scale
        NewElement.CoordinatesY(1)=width*Scale
        If i>0 Then
            Set NewElement=ob.AddElement("EPC_SF_Label",CSng(Scale*i-
15),CSng(Scale*width+5))
            NewElement!Value=i & " m"
        End If
    Next
End Sub

Public Sub EPC_SF_Shop_DrawLayout(ob As CFCSim_ModelingElementInstance, Length As
Integer,width As Integer,GridIncrement As Integer,Scale As Integer)
    Dim NewElement As CFCSim_ModelingElementInstance
    Dim i As Integer

    For Each NewElement In ob.ChildElements
        If NewElement.ElementType="EPC_SF_Layout" Then
            NewElement.Delete
        End If
    Next

    If Not IsNull(ob("DWG")) And Not IsEmpty(ob("DWG")) Then
        Set NewElement=ob.AddElement("EPC_SF_Layout",1,1)
        NewElement.CoordinatesX(0)=0
        NewElement.CoordinatesY(0)=0
        NewElement.CoordinatesX(1)=Length*Scale
        NewElement.CoordinatesY(1)=width*Scale
    End If

End Sub

Public Sub EPC_SF_Shop_OnDragDraw(ob As CFCSim_ModelingElementInstance)
    CDC.Rectangle
ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(0)+ob!Length*ob.Parent("Scale"),ob.Coordinat
esY(0)+ob!width*ob.Parent("Scale")
    ob("X")=ob.CoordinatesX(0)/ob.Parent("Scale")
    ob("Y")=ob.CoordinatesY(0)/ob.Parent("Scale")
End Sub

Public Sub EPC_SF_Shop_OnDraw(ob As CFCSim_ModelingElementInstance)
    If ob.Selected Then
        CDC.ChangeLineStyle CFC_SOLID,1,RGB(255,0,0)
    End If
ob("ShopDWG"),ob.CoordinatesX(0),ob.CoordinatesY(0),ob("Length"),ob("Width")
    CDC.Rectangle
ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(0)+ob!Length*ob.Parent("Scale"),ob.Coordinat
esY(0)+ob!width*ob.Parent("Scale")
    CDC.TextOut ob.CoordinatesX(0)+10,ob.CoordinatesY(0)+10,ob("Shop")
    CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)
    ob.DrawConnectionPoints
    If ob("Scale")<>0 Then EPC_SF_Shop_DrawGrid
ob,ob!Length,ob!width,ob!GridIncrement,ob!Scale
```

```
End Sub

Public Sub EPC_SF_Shop_OnGetBoundingRect(ob As CFCSim_ModelingElementInstance, mRect As
CFCGraphics_Rect, HitTest As Boolean)
        mRect.Left=ob.CoordinatesX(0)
        mRect.Right=ob.CoordinatesX(0)+ob!Length*ob.Parent("Scale")
        mRect.top=ob.CoordinatesY(0)
        mRect.bottom=ob.CoordinatesY(0)+ob!width*ob.Parent("Scale")

        Dim cp As CFCSim_ConnectionPoint
        For Each cp In ob.ConnectionPoints
                mRect.UnionPoint cp.x-cp.Tolerance, cp.y-cp.Tolerance
                mRect.UnionPoint cp.x+cp.Tolerance, cp.y-cp.Tolerance
                mRect.UnionPoint cp.x-cp.Tolerance, cp.y+cp.Tolerance
                mRect.UnionPoint cp.x+cp.Tolerance, cp.y+cp.Tolerance
        Next

End Sub

Public Function EPC_SF_Shop_OnValidateParameters(ob As CFCSim_ModelingElementInstance,
Parameters As Object) As Boolean
        EPC_SF_Shop_OnValidateParameters=ob.OnValidateParameters(Parameters,True)
        If Not EPC_SF_Shop_OnValidateParameters Then Exit Function

        EPC_SF_Shop_DrawGrid
ob,Parameters("Length"),Parameters("Width"),Parameters("GridIncrement"),Parameters("Scale")

        ob.RefreshDisplay
End Function
```

EPC_SF_WorkCell

```
Public Function EPC_SF_WorkCell_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y
As Single) As Boolean
        ob.OnCreate x,y,True
        EPC_SF_WorkCell_OnCreate=True

        ob.AddAttribute "Description","WorkCell Name",CFC_Text, CFC_ListBox, CFC_ReadWrite
        ob.AddAttribute "Capacity","WorkCell Capacity (Number of Entities / Accumulative Value of
Entity Attribute)",CFC_Text, CFC_ListBox, CFC_ReadWrite
        ob.AddAttribute "Measure","Capacity Measured By (An Entity Attribute / Leave It Blank If By
Number)",CFC_Text, CFC_Single, CFC_ReadWrite

        ob!Description="Bay1-FW-WC1"
        ob!Length=5
        ob!width=5
        ob("Capacity")="Unconstrained"
        ob("Measure")=""

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+75
        ob.CoordinatesY(1)=y+75
        ob.AddResource "CapacityRes", 1  'CInt(ob("Capacity"))

End Function

Public Sub EPC_SF_WorkCell_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub EPC_SF_WorkCell_OnDraw(ob As CFCSim_ModelingElementInstance)
        Dim d1, d2 As Integer
        CDC.ChangeFont "Courier New",13,True,False,False,False

        CDC.RenderPicture "WorkCell.bmp",ob.CoordinatesX(0)+1,ob.CoordinatesY(0)+1,32,32
        CDC.Rectangle
ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(0)+33,ob.CoordinatesY(0)+33
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)

        CDC.ChangeFont "Arial",11,True,False,False,False
        CDC.TextOut ob.CoordinatesX(0)+5,ob.CoordinatesY(0)+40,ob("Description")

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        ob.DrawConnectionPoints

End Sub

Public Sub EPC_SF_WorkCell_OnListBoxInitialize(ob As CFCSim_ModelingElementInstance, attr As
CFCSim_Attribute, lstList As Object)
        Select Case attr.Name
                Case "Description"
                        lstList.additem "Bay1-Cutting"
                        lstList.additem "Bay1-FW-WC1"
                        lstList.additem "Bay1-FW-WC2"
                        lstList.additem "Bay1-FW-WC3"
                        lstList.additem "Bay1-FW-WC4"
                        lstList.additem "Bay1-FW-WC5"
                        lstList.additem "Bay1-FW-WC6"

                        lstList.additem "Bay2-Cutting"
                        lstList.additem "Bay2-FW-WC1"
                        lstList.additem "Bay2-FW-WC2"
                        lstList.additem "Bay2-FW-WC3"
                        lstList.additem "Bay2-FW-WC4"
                        lstList.additem "Bay2-FW-WC5"
                        lstList.additem "Bay2-FW-WC6"

                        lstList.additem "Bay3-Cutting"
                        lstList.additem "Bay3-FW-WC1"
                        lstList.additem "Bay3-FW-WC2"
                        lstList.additem "Bay3-FW-WC3"
                        lstList.additem "Bay3-FW-WC4"
                        lstList.additem "Bay3-FW-WC5"
```

```
            lstList.additem "Bay3-FW-WC6"

            lstList.additem "Bay4-Cutting"
            lstList.additem "Bay4-FW-WC1"
            lstList.additem "Bay4-FW-WC2"
            lstList.additem "Bay4-FW-WC3"
            lstList.additem "Bay4-FW-WC4"
            lstList.additem "Bay4-FW-WC5"
            lstList.additem "Bay4-FW-WC6"

            lstList.additem "Bay5-Cutting"
            lstList.additem "Bay5-FW-WC1"
            lstList.additem "Bay5-FW-WC2"
            lstList.additem "Bay5-FW-WC3"
            lstList.additem "Bay5-FW-WC4"
            lstList.additem "Bay5-FW-WC5"
            lstList.additem "Bay5-FW-WC6"
            lstList.additem "Bay5-HT"

            lstList.additem "SR-1"
            lstList.additem "SR-2"

            lstList.additem "HT"

            lstList.additem "Painting1-Rack1"
            lstList.additem "Painting1-Rack2"
            lstList.additem "Painting2-Rack1"
            lstList.additem "Painting2-Rack2"

        Case "Capacity"
            lstList.additem "Unconstrained"
    End Select

End Sub

Public Sub EPC_SF_WorkCell_OnSimulationInitialize(ob As CFCSim_ModelingElementInstance)
    If ob("capacity")<>"Unconstrained" Then
        ob.res("CapacityRes").NumResources=CInt(ob("Capacity"))
    End If

    ob.AddEvent "TransferIn",True
    ob.AddEvent "ReleaseCapacity"

End Sub

Public Sub EPC_SF_WorkCell_OnSimulationProcessEvent(ob As CFCSim_ModelingElementInstance,
MyEvent As String, Entity As CFCSim_Entity)
    Dim Ele As CFCSim_ModelingElementInstance

    Select Case MyEvent
        Case "TransferIn"
            For Each Ele In ob.ChildElements
                If Ele.ElementType="InPort" Then
                    ob.TransferOut Entity, ob.ConnectionPoints("In1")
                    Exit Sub
                End If
            Next
            'Tracer.Trace "After capture, Number of Resources is " &
ob.res("capacityres").NumBusyResources

        Case "ReleaseCapacity"
            '********Release capacity when entity is transfered out from the
workcell******
            If ob("capacity")="Unconstrained" Then     'It means capacity of workcell is
Unconstrained
                ob.TransferOut Entity, ob.ConnectionPoints(Entity("OutPort"))
                '(Transfer out the entity to the specific point. Otherwise it is cloned and
transfered out to all points)
                Exit Sub
            End If

            If ob("measure")="" Then 'It means capacity of workcell is measured only by
number of entities
                ob.ReleaseResource "CapacityRes", Entity
                ob.TransferOut Entity, ob.ConnectionPoints(Entity("OutPort"))
            Else     'It means capacity of workcell is measured by one of attributes of entity
                ob.ReleaseResource "CapacityRes", Entity, Entity(ob("Measure"))
                ob.TransferOut Entity, ob.ConnectionPoints(Entity("OutPort"))
            End If
            '********Release capacity when entity is transfered out from the
workcell******
    End Select

End Sub

Public Sub EPC_SF_WorkCell_OnSimulationTransferIn(ob As CFCSim_ModelingElementInstance, Entity
As CFCSim_Entity, srcCP As CFCSim_ConnectionPoint, dstCP As CFCSim_ConnectionPoint)

    '********To determine if it is a TransferIn or TransferOut.************
    '$$$Actually TransferOut is a type of TransterIn (Refer to OnSimulationTransferIn to understand
it).$$$
    If srcCP.ModelingElement.ElementType = "OutPort" Then
        Entity("OutPort")=srcCP.RelationsTo(1).dstConnection.Name 'Remember which
OutPort, from which the entity transfers out.
        ob.ScheduleEvent Entity,"ReleaseCapacity",0.0
    Else
        ob.ScheduleEvent Entity,"TransferIn",0.0
    End If
    '********To determine if it is a TransferIn or TransferOut.*************

End Sub

Public Function EPC_SF_WorkCell_OnValidateParameters(ob As CFCSim_ModelingElementInstance,
Parameters As Object) As Boolean
    EPC_SF_WorkCell_OnValidateParameters=ob.OnValidateParameters(Parameters,True)
    If Not EPC_SF_WorkCell_OnValidateParameters Then Exit Function
End Function
```

EPC_SF_LayDown

```
Public Function EPC_SF_LayDown_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y As Single) As Boolean
        ob.OnCreate x,y,True
        EPC_SF_LayDown_OnCreate=True

        ob.AddAttribute "Description","LayDown Name",CFC_Text, CFC_ListBox, CFC_ReadWrite
        ob.AddAttribute "Length","LayDown Length in meters",CFC_Numeric, CFC_Single, CFC_ReadWrite,0,5000
        ob.AddAttribute "Width","LayDown Width in meters",CFC_Numeric, CFC_Single, CFC_ReadWrite,1,5000
        ob.AddAttribute "Capacity","LayDown Capacity (Number of Entities / Accumulative Value of Entity Attribute)",CFC_Text, CFC_ListBox, CFC_ReadWrite
        ob.AddAttribute "Measure","Capacity Measured By (Entity Attribute / Leave It Blank If By Number)",CFC_Text, CFC_Single, CFC_ReadWrite

        ob!Description="Bay1-PipesLD"
        ob!Length=5
        ob!width=5
        ob("Capacity")="Unconstrained"
        ob("Measure")=""

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+75          'ob!Length*ob.Parent("Scale")
        ob.CoordinatesY(1)=y+75          'ob!width*ob.Parent("Scale")

        ob.AddResource "CapacityRes", 1            'CInt(ob("Capacity"))

End Function

Public Sub EPC_SF_LayDown_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub EPC_SF_LayDown_OnDraw(ob As CFCSim_ModelingElementInstance)
        Dim d1, d2 As Integer

        CDC.ChangeFont "Courier New",13,True,False,False,False

        CDC.RenderPicture "LayDown.bmp",ob.CoordinatesX(0)+1,ob.CoordinatesY(0)+1,32,32
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(0)+33,ob.CoordinatesY(0)+33
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)

        CDC.ChangeFont "Arial",11,True,False,False,False
        CDC.TextOut ob.CoordinatesX(0)+5,ob.CoordinatesY(0)+40,ob("Description")

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        ob.DrawConnectionPoints

End Sub
```

```
Public Sub EPC_SF_LayDown_OnListBoxInitialize(ob As CFCSim_ModelingElementInstance, attr As
CFCSim_Attribute, lstList As Object)
        Select Case attr.Name
                Case "Description"
                lstList.additem "Bay1-PipesLD"
                lstList.additem "Bay1-SpoolsLD"

                lstList.additem "Bay2-PipesLD"
                lstList.additem "Bay2-SpoolsLD"

                lstList.additem "Bay3-PipesLD"
                lstList.additem "Bay3-SpoolsLD"

                lstList.additem "Bay4-PipesLD"
                lstList.additem "Bay4-SpoolsLD"

                lstList.additem "Bay5-PipesLD"
                lstList.additem "Bay5-SpoolsLD"

                lstList.additem "Yard-SpoolsLD"

                Case "Capacity"
                lstList.additem "Unconstrained"
        End Select
End Sub

Public Sub EPC_SF_LayDown_OnSimulationInitialize(ob As CFCSim_ModelingElementInstance)
        If ob("capacity")<>"Unconstrained" Then
                ob.res("CapacityRes").NumResources=CInt(ob("Capacity"))
        End If

        ob.AddEvent "TransferIn",True
        ob.AddEvent "ReleaseCapacity"

End Sub

Public Sub EPC_SF_LayDown_OnSimulationProcessEvent(ob As CFCSim_ModelingElementInstance,
MyEvent As String, Entity As CFCSim_Entity)
        Dim Ele As CFCSim_ModelingElementInstance

        Select Case MyEvent
                Case "TransferIn"
                        For Each Ele In ob.ChildElements
                                If Ele.ElementType="InPort" Then
                                        'Just transfer the entity into the child window.
                                        ob.TransferOut Entity, ob.ConnectionPoints("In1")
                                        'Operation of resource capturing is done in Element
"CapacityDetector"
                                        Exit Sub
                                End If
                        Next
                        'Tracer.Trace "After capture, Number of Resources is " &
ob.res("capacityres").NumBusyResources

                Case "ReleaseCapacity"
```

```
                    '********Release capacity when entity is transfered out from the LD******
                    If ob("capacity")="Unconstrained" Then     'It means capacity of workcell is
Unconstrained
                        ob.TransferOut Entity, ob.ConnectionPoints(Entity("OutPort"))
                        '(Transfer out the entity to the specific point. Otherwise it is cloned and
transfered out to all points)
                        Exit Sub
                    End If

                    If ob("measure")="" Then 'It means capacity of workcell is measured only by
number of entities
                        ob.ReleaseResource "CapacityRes", Entity
                        ob.TransferOut Entity, ob.ConnectionPoints(Entity("OutPort"))
                    Else    'It means capacity of workcell is measured by one of attributes of entity
                        ob.ReleaseResource "CapacityRes", Entity, Entity(ob("Measure"))
                        ob.TransferOut Entity, ob.ConnectionPoints(Entity("OutPort"))
                    End If
                    '********Release capacity when entity is transfered out from the LD******
    End Select

End Sub

Public Sub EPC_SF_LayDown_OnSimulationTransferIn(ob As CFCSim_ModelingElementInstance, Entity
As CFCSim_Entity, SrcCp As CFCSim_ConnectionPoint, DstCp As CFCSim_ConnectionPoint)

    '********To determine if it is a TransferIn or TransferOut.************
    '$$$Actually TransferOut is a type of TransterIn (Refer to OnSimulationTransferIn to understand
it).$$$
    If SrcCp.ModelingElement.ElementType = "OutPort" Then   'It is actually TransferOut
        Entity("OutPort")=SrcCp.RelationsTo(1).dstConnection.Name 'Remember which
OutPort, from which the entity transfers out.
        'Tracer.Trace "Outport Name is " & Entity("OutPort")
        ob.ScheduleEvent Entity,"ReleaseCapacity",0.0
    Else
        'It is a real TransferIn
        ob.ScheduleEvent Entity,"TransferIn",0.0
    End If
    '********To determine if it is a TransferIn or TransferOut.************

End Sub

Public Function EPC_SF_LayDown_OnValidateParameters(ob As CFCSim_ModelingElementInstance,
Parameters As Object) As Boolean
    EPC_SF_LayDown_OnValidateParameters=ob.OnValidateParameters(Parameters,True)
    If Not EPC_SF_LayDown_OnValidateParameters Then Exit Function
End Function
```

EPC_SF_ProcessControler

```
Public Function EPC_SF_ProcessControler_OnCreate(ob As CFCSim_ModelingElementInstance, x As
Single, y As Single) As Boolean
    EPC_SF_ProcessControler_OnCreate=True
    ob.OnCreate x,y,True
```

```
        ob.AddAttribute "NextProcess", "Next Process to Go
Through",CFC_Text,CFC_ListBox,CFC_ReadWrite
        ob("NextProcess")="Cut"

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+35
        ob.CoordinatesY(1)=y+35

        ob.AddConnectionPoint "IN",x-25,y,CInput,5,False
        ob.AddConnectionPoint "Yes",x+25,y-25,COutput,5,False
  ob.AddConnectionPoint "No",x+25,y+25,COutput,5,False

End Function

Public Sub EPC_SF_ProcessControler_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub EPC_SF_ProcessControler_OnDraw(ob As CFCSim_ModelingElementInstance)
        Dim x As Single
        Dim y As Single

        CDC.RenderPicture "ProcessControler.bmp",ob.CoordinatesX(0)-24,ob.CoordinatesY(0)-
24,32,32
        CDC.ChangeLineStyle CFC_SOLID,2,RGB(0,0,0)
        CDC.MoveTo ob.CoordinatesX(0),ob.CoordinatesY(0)
        CDC.LineTo      ob.ConnectionPoints("Yes").x,ob.ConnectionPoints("Yes").y

        CDC.Arrow
ob.ConnectionPoints("In").x,ob.ConnectionPoints("In").y,ob.CoordinatesX(0),ob.CoordinatesY(0),7
        CDC.MoveTo ob.CoordinatesX(0),ob.CoordinatesY(0)
        CDC.LineTo ob.ConnectionPoints("No").x , ob.ConnectionPoints("No").y

        CDC.ChangeFont "Arial",12,True,False,False,False
        x=(ob.CoordinatesX(0)+ob.ConnectionPoints("Yes").x )/2-5
        y=(ob.CoordinatesY(0)+ob.ConnectionPoints("Yes").y)/2-5
        CDC.TextOut x ,y , "Yes"
        x=(ob.CoordinatesX(0)+ob.ConnectionPoints("No").x )/2-5
        y=(ob.CoordinatesY(0)+ob.ConnectionPoints("No").y)/2-5
        CDC.TextOut x,y, "No"

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-30,ob.CoordinatesY(0)-
30,ob.CoordinatesX(1),ob.CoordinatesY(1)
        End If

  ob.DrawConnectionPoints
End Sub

Public Sub EPC_SF_ProcessControler_OnListBoxInitialize(ob As CFCSim_ModelingElementInstance,
attr As CFCSim_Attribute, lstList As Object)
        lstList.additem "Cutting"
        lstList.additem "FittingWelding"
```

```
    lstList.additem "RT"
    lstList.additem "LT"
    lstList.additem "MT"
    lstList.additem "HT"
    lstList.additem "StressRelief"
    lstList.additem "HydroTest"
    lstList.additem "Painting"

End Sub

Public Sub EPC_SF_ProcessControler_OnMove(ob As CFCSim_ModelingElementInstance, ByVal x1 As
Single, ByVal y1 As Single, ByVal x2 As Single, ByVal y2 As Single)
    Dim Ang As Double
    Dim xo As Double
    Dim yo As Double
    Dim cp As CFCSim_ConnectionPoint

    Dim angle1 As Double
    Dim angle2 As Double

    If Sqr((ob.CoordinatesX(0) - x1) ^ 2 + (ob.CoordinatesY(0) - y1) ^ 2) > 20  Then

        angle1=EPC_SF_ProcessControler_GetAngle(x1-ob.CoordinatesX(0)
,ob.CoordinatesY(0)-y1)
        angle2 =EPC_SF_ProcessControler_GetAngle(x2-ob.CoordinatesX(0)
,ob.CoordinatesY(0)-y2)

        Ang=angle1-angle2

        For Each cp In ob.ConnectionPoints
            xo=cp.x - ob.CoordinatesX(0)
            yo=- ( cp.y - ob.CoordinatesY(0))
            cp.x = xo*Cos(Ang)+yo*Sin(Ang) +ob.CoordinatesX(0)
            cp.y = -(-xo*Sin(Ang)+yo*Cos(Ang)) + ob.CoordinatesY(0)
        Next
    Else
        ' call default on move
        ob.OnMove x1,y1,x2,y2, True
    End If
End Sub
Public Function  EPC_SF_ProcessControler_GetAngle(ByVal x As Single, ByVal y As Single) As Single
        If x>0 And y>0 Then
            EPC_SF_ProcessControler_GetAngle=Atn(y/x)
        ElseIf x<0 And y>0 Then
            EPC_SF_ProcessControler_GetAngle=3.14159-Atn(Abs(y/x))
        ElseIf x<0 And y<0 Then
            EPC_SF_ProcessControler_GetAngle=3.14159+Atn(Abs(y/x))
        ElseIf x>0 And y<0 Then
            EPC_SF_ProcessControler_GetAngle=2*3.14159-Atn(Abs(y/x))
        Else
            EPC_SF_ProcessControler_GetAngle=-1
        End If
End Function
```

```
Public Sub EPC_SF_ProcessControler_OnSimulationTransferIn(ob As
CFCSim_ModelingElementInstance, Entity As CFCSim_Entity, SrcCp As CFCSim_ConnectionPoint,
DstCp As CFCSim_ConnectionPoint)

        If Entity(ob("NextProcess"))="Y" Then
                ob.TransferOut Entity,ob.ConnectionPoints("Yes")
        Else
                ob.TransferOut Entity,ob.ConnectionPoints("No")
        End If

End Sub
```

EPC_SF_DispatchControler

```
Public Function EPC_SF_DispatchControler_OnCreate(ob As CFCSim_ModelingElementInstance, x As
Single, y As Single) As Boolean
        ob.OnCreate x,y,True
        EPC_SF_DispatchControler_OnCreate=True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+50
        ob.CoordinatesY(1)=y+100

        ob.AddConnectionPoint "In" , x-5, y+50, CInput, 5

        ob.AddAttribute "DispatchCriteria","Dispatch Criteria (An Attribute of Spool)",CFC_Text,
CFC_Single, CFC_ReadWrite
        ob("DispatchCriteria")="Length"
        Dim i As Integer
        For i=1 To 6
                ob.AddAttribute "Dispatch" & i,"Transfer out from Out" & i & " if Dispatch Criteria is
<=",CFC_Numeric, CFC_Single, CFC_ReadWrite
                ob.AddConnectionPoint "Out" & i,x+55,y+12*i,COutput,5
        Next

End Function

Public Sub EPC_SF_DispatchControler_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub EPC_SF_DispatchControler_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.ChangeFont "Courier New",13,True,False,False,False
        CDC.RenderPicture "ProcessControler.bmp",ob.CoordinatesX(0),ob.CoordinatesY(0),32,32
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)

        Dim i As Integer
        For i=1 To 6
                CDC.TextOut ob.CoordinatesX(0)+25,ob.CoordinatesY(0)+12*i-6, "Out" & i
        Next

        If ob.Selected Then
        CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
        CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
    End If
    CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)

    ob.DrawConnectionPoints
End Sub

Public Sub EPC_SF_DispatchControler_OnSimulationTransferIn(ob As
CFCSim_ModelingElementInstance, Entity As CFCSim_Entity, SrcCp As CFCSim_ConnectionPoint,
DstCp As CFCSim_ConnectionPoint)
    Dim i As Integer

    For i=1 To 6
        Tracer.Trace "-----Length of Entity " & Entity.Id & " is " &
Entity(ob("DispatchCriteria"))
        Tracer.Trace "Dispatch " & i & " is " & ob("Dispatch" & i)

        If CInt(Entity(ob("DispatchCriteria")))<=CInt(ob("Dispatch" & i)) Then
            Tracer.Trace "Entity " & Entity.Id & " is transfered out from Out " & i
            ob.TransferOut Entity, ob.ConnectionPoints("Out" & i)
            Exit Sub
        End If
    Next

End Sub
```

EPC_SF_CapacityDetector

```
Public Function EPC_SF_CapacityDetector_OnCreate(ob As CFCSim_ModelingElementInstance, x As
Single, y As Single) As Boolean
    ob.OnCreate x,y,True
    EPC_SF_CapacityDetector_OnCreate=True

    ob.SetNumCoordinates 2
    ob.CoordinatesX(0)=x
    ob.CoordinatesY(0)=y
    ob.CoordinatesX(1)=x+50
    ob.CoordinatesY(1)=y+50

    ob.AddAttribute "Destination", "Name of Destination Location To Detect",
CFC_Text,CFC_ListBox, CFC_ReadWrite
    ob.AddAttribute "DestinationType", "Tyep of Destination Location To Detect",
CFC_Text,CFC_ListBox, CFC_ReadWrite
    ob("DestinationType")="EPC_SF_WorkCell"
    ob("Destination")="Bay1-FW-WC1"

    ob.AddConnectionPoint "In" , x-5, y+25, CInput, 5
    ob.AddConnectionPoint "Out",x+55,y+25,COutput,5

End Function
```

```
Public Sub EPC_SF_CapacityDetector_OnListBoxInitialize(ob As CFCSim_ModelingElementInstance,
attr As CFCSim_Attribute, lstList As Object)
        Select Case attr.Name
                Case "DestinationType"
                        lstList.additem "EPC_SF_WorkCell"
                        lstList.additem "EPC_SF_LayDown"

                Case "Destination"
                                                lstList.additem "Bay1-Cutting"
                                                lstList.additem "Bay1-FW-WC1"
                                                lstList.additem "Bay1-FW-WC2"
                                                lstList.additem "Bay1-FW-WC3"
                                                lstList.additem "Bay1-FW-WC4"
                                                lstList.additem "Bay1-FW-WC5"
                                                lstList.additem "Bay1-FW-WC6"

                                                lstList.additem "Bay2-Cutting"
                                                lstList.additem "Bay2-FW-WC1"
                                                lstList.additem "Bay2-FW-WC2"
                                                lstList.additem "Bay2-FW-WC3"
                                                lstList.additem "Bay2-FW-WC4"
                                                lstList.additem "Bay2-FW-WC5"
                                                lstList.additem "Bay2-FW-WC6"

                                                lstList.additem "Bay3-Cutting"
                                                lstList.additem "Bay3-FW-WC1"
                                                lstList.additem "Bay3-FW-WC2"
                                                lstList.additem "Bay3-FW-WC3"
                                                lstList.additem "Bay3-FW-WC4"
                                                lstList.additem "Bay3-FW-WC5"
                                                lstList.additem "Bay3-FW-WC6"

                                                lstList.additem "Bay4-Cutting"
                                                lstList.additem "Bay4-FW-WC1"
                                                lstList.additem "Bay4-FW-WC2"
                                                lstList.additem "Bay4-FW-WC3"
                                                lstList.additem "Bay4-FW-WC4"
                                                lstList.additem "Bay4-FW-WC5"
                                                lstList.additem "Bay4-FW-WC6"

                                                lstList.additem "Bay5-Cutting"
                                                lstList.additem "Bay5-FW-WC1"
                                                lstList.additem "Bay5-FW-WC2"
                                                lstList.additem "Bay5-FW-WC3"
                                                lstList.additem "Bay5-FW-WC4"
                                                lstList.additem "Bay5-FW-WC5"
                                                lstList.additem "Bay5-FW-WC6"
                                                lstList.additem "Bay5-HT"

                                                lstList.additem "SR-1"
                                                lstList.additem "SR-2"

                                                lstList.additem "HT"

                                                lstList.additem "Painting1-Rack1"
                                                lstList.additem "Painting1-Rack2"
```

```
                                        lstList.additem "Painting2-Rack1"
                                        lstList.additem "Painting2-Rack2"

                                        lstList.additem "Bay1-PipesLD"
                                        lstList.additem "Bay1-SpoolsLD"
                                        lstList.additem "Bay2-PipesLD"
                                        lstList.additem "Bay2-SpoolsLD"
                                        lstList.additem "Bay3-PipesLD"
                                        lstList.additem "Bay3-SpoolsLD"
                                        lstList.additem "Bay4-PipesLD"
                                        lstList.additem "Bay4-SpoolsLD"
                                        lstList.additem "Bay5-PipesLD"
                                        lstList.additem "Bay5-SpoolsLD"
                                        lstList.additem "Yard-SpoolsLD"
        End Select
End Sub

Public Sub EPC_SF_CapacityDetector_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub EPC_SF_CapacityDetector_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.RenderPicture "CapacityDetector.bmp",ob.CoordinatesX(0),ob.CoordinatesY(0),25,25
        CDC.Rectangle
ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(0)+26,ob.CoordinatesY(0)+26
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)

        CDC.ChangeFont "Arial",11,True,False,False,False
        CDC.TextOut ob.CoordinatesX(0)+5,ob.CoordinatesY(0)+27, "Capacity"
        CDC.TextOut ob.CoordinatesX(0)+5,ob.CoordinatesY(0)+38, "Detector"

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)

        ob.DrawConnectionPoints
End Sub

Public Sub EPC_SF_CapacityDetector_OnSimulationInitialize(ob As CFCSim_ModelingElementInstance)
        ob.AddEvent "RequestCapacity",True

End Sub

Public Sub EPC_SF_CapacityDetector_OnSimulationProcessEvent(ob As
CFCSim_ModelingElementInstance, MyEvent As String, Entity As CFCSim_Entity)
        Dim Des As CFCSim_ModelingElementInstance
        Dim i As Long

        Select Case MyEvent
                Case "RequestCapacity"
                        'Locate Destination element
                        For Each Des In Elements
                                If Des.ElementType=ob("DestinationType") Then
```

```
                    If Des("Description")=ob("Destination") Then

                        '********If capacity of destination is available, send
entity out******
                        If Des("capacity")="Unconstrained" Then    'It
means capacity of destination is Unconstrained
                            ob.TransferOut Entity,
ob.ConnectionPoints("Out")
                            Exit Sub
                        End If

                        If Des("measure")="" Then            'It means
capacity of workcell is measured only by number of entities
                            If Des.RequestResource("CapacityRes",
Entity) Then
                                ob.TransferOut Entity,
ob.ConnectionPoints("Out")
                                Exit Sub
                        End If
                        Else      'It means capacity of workcell is measured
by one of attributes of entity
                            If Des.RequestResource("CapacityRes",
Entity, Entity(Des("Measure"))) Then
                                ob.TransferOut Entity,
ob.ConnectionPoints("Out")
                                Exit Sub
                            End If
                        End If
                        '********If capacity of destination is available, send
entity out******

                    End If
                End If
            Next

    End Select
End Sub
```

EPC_SF_Resource

```
Public Sub EPC_SF_Resource_OnAfterUpdateParameters(ob As CFCSim_ModelingElementInstance)
    Elements(CStr(ob("ResID")))("ResName")=ob("ResName")
    Elements(CStr(ob("ResID")))("Total")=ob("Total")
    Elements(CStr(ob("FileID")))("FileName")=ob("ResName") & "Q"
End Sub

Public Function EPC_SF_Resource_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y
As Single) As Boolean
    EPC_SF_Resource_OnCreate=True
    'ob.OnCreate x,y,True
  ob.SetNumCoordinates 2
    ob.CoordinatesX(0)=x
    ob.CoordinatesY(0)=y
    ob.CoordinatesX(1)=x+10
```

```
          ob.CoordinatesY(1)=y+10

          ob.AddAttribute "ResName","Resource Description",CFC_Text, CFC_Single,CFC_ReadWrite
   ob.AddAttribute "Total","Total Number of Resources",CFC_Numeric,CFC_Single,
CFC_ReadWrite,0,1000000
          ob.AddAttribute "Length","Resource Length",CFC_Numeric, CFC_Single,
CFC_ReadWrite,1,5000
          ob.AddAttribute "Width","Resource Width",CFC_Numeric, CFC_Single, CFC_ReadWrite,1,5000
   ob.AddAttribute "Current","Current Number of Available Resources",CFC_Numeric, CFC_Single,
CFC_ReadOnly

          ob.AddAttribute "ResID","",CFC_Numeric, CFC_Single, CFC_Hidden
          ob.AddAttribute "FileID","",CFC_Numeric, CFC_Single, CFC_Hidden
          ob!Length=20
          ob!width=50

          ob("ResName")="Res" & ob.Id

          Dim NewRes, NewFile As CFCSim_ModelingElementInstance
          'Add default elements and relations
          Set NewRes=ob.AddElement("Resource",10,10)
          ob("ResID")=NewRes.Id
          Set NewFile=ob.AddElement("Waiting_File",80,10)
          ob("FileID")=NewFile.Id

End Function

Public Sub EPC_SF_Resource_OnDragDraw(ob As CFCSim_ModelingElementInstance)
          CDC.Rectangle
ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(0)+ob!Length*ob.Parent("Scale"),ob.Coordinat
esY(0)+ob!width*ob.Parent("Scale")
End Sub

Public Sub EPC_SF_Resource_OnDraw(ob As CFCSim_ModelingElementInstance)
          Dim d1, d2 As Integer
          CDC.ChangeLineStyle  CFC_SOLID, 1,RGB(120,180,0)
          CDC.TextOut ob.CoordinatesX(0)+2,ob.CoordinatesY(0)+2,ob("ResName")
          d1=ob.CoordinatesX(0)+ob!Length
          d2=ob.CoordinatesY(0)+ob!width
          CDC.Rectangle
ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(0)+ob!Length,ob.CoordinatesY(0)+ob!width

          If ob.Selected Then
                    CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                    CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-2,d1+2,d2+2
          End If
          CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)
          'ob.DrawConnectionPoints
End Sub

Public Sub EPC_SF_Resource_OnGetBoundingRect(ob As CFCSim_ModelingElementInstance, mRect As
CFCGraphics_Rect, HitTest As Boolean)
          mRect.Left=ob.CoordinatesX(0)-5
          mRect.Right=ob.CoordinatesX(0)+ob!Length+5
          mRect.top=ob.CoordinatesY(0)-5
          mRect.bottom=ob.CoordinatesY(0)+ob!width+5
```

```
End Sub

Public Function EPC_SF_Resource_OnValidateParameters(ob As CFCSim_ModelingElementInstance,
Parameters As Object) As Boolean
        EPC_SF_Resource_OnValidateParameters=True
End Function
```

EPC_SF_Path

```
Public Function EPC_SF_Path_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y As
Single) As Boolean
        EPC_SF_Path_OnCreate=True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+50
        ob.CoordinatesY(1)=y+50

        ob.AddAttribute "Duration", "Movement Duration in Minutes", CFC_Numeric, CFC_Single,
CFC_ReadWrite,


        ob.AddConnectionPoint "c1", ob.CoordinatesX(0),ob.CoordinatesY(0),CInput,5
        ob.AddConnectionPoint "c2", ob.CoordinatesX(1),ob.CoordinatesY(1),COutput,5

End Function

Public Sub EPC_SF_Path_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub EPC_SF_Path_OnDraw(ob As CFCSim_ModelingElementInstance)
        Dim ratio As Double
      Dim Length As Integer
      Dim xoffset As Integer
      Dim yoffset As Integer
      Dim x As Single
      Dim y As Single

        ob.DrawConnectionPoints

        With CDC
            Length = Sqr((ob.CoordinatesX(1) - ob.CoordinatesX(0)) ^ 2 + (ob.CoordinatesY(1) -
ob.CoordinatesY(0)) ^ 2)
        If Length<5 Then Exit Sub
        ratio = Length/ 5
            If ob.Selected Then
                CDC.ChangeLineStyle CFC_SOLID, 1,RGB(255,0,0)
            End If

          xoffset = (ob.CoordinatesY(1) - ob.CoordinatesY(0)) / ratio
          yoffset = (ob.CoordinatesX(1) - ob.CoordinatesX(0) )/ ratio
      .MoveTo ob.CoordinatesX(0) - xoffset, ob.CoordinatesY(0)+ yoffset
      .LineTo ob.CoordinatesX(1) - xoffset, ob.CoordinatesY(1) + yoffset

        .MoveTo ob.CoordinatesX(0) + xoffset, ob.CoordinatesY(0) - yoffset
      .LineTo  ob.CoordinatesX(1) + xoffset, ob.CoordinatesY(1) - yoffset

      .ArrowHead      ob.CoordinatesX(0)          ,ob.CoordinatesY(0)      ,      ob.CoordinatesX(1)
,ob.CoordinatesY(1) ,10

    End With

End Sub

Public Sub EPC_SF_Path_OnGetBoundingRect(ob As CFCSim_ModelingElementInstance, mRect As
CFCGraphics_Rect, HitTest As Boolean)

    Dim t As Single
        t=5

    If ob.CoordinatesX(0)<ob.CoordinatesX(1) Then
        mRect.Left=ob.CoordinatesX(0)-t
        mRect.Right=ob.CoordinatesX(1)+t
    Else
        mRect.Left=ob.CoordinatesX(1)-t
        mRect.Right=ob.CoordinatesX(0)+t
    End If
    If Abs(ob.CoordinatesX(0)-ob.CoordinatesX(1))<=5 Then
        mRect.Left=ob.CoordinatesX(1)-10
        mRect.Right=ob.CoordinatesX(1)+10
    End If
    If ob.CoordinatesY(0)<ob.CoordinatesY(1) Then
        mRect.top=ob.CoordinatesY(0)-t
        mRect.bottom=ob.CoordinatesY(1)+t
    Else
        mRect.top=ob.CoordinatesY(1)-t
        mRect.bottom=ob.CoordinatesY(0)+t
    End If
    If Abs(ob.CoordinatesY(0)-ob.CoordinatesY(1))<=5 Then
        mRect.top=ob.CoordinatesY(1)-10
        mRect.bottom=ob.CoordinatesY(1)+10
    End If

End Sub

Public Function EPC_SF_Path_OnHitTest(ob As CFCSim_ModelingElementInstance, x As Single, y As
Single) As Boolean
    Dim slope As Double
    Dim Temp1 As Double
    Dim Temp2 As Double

    Dim mRect As New CFCGraphics_Rect
    Dim InRect As Boolean

    EPC_SF_Path_OnHitTest=False
```

```
        'first check if point is inside the bounding rectangle
        ob.OnGetBoundingRect mRect,True
        If Not (x>mRect.Left And x<mRect.Right And y>mRect.top And y<mRect.bottom) Then
            Exit Function
        End If

        ' Then check some special cases

        If    (Abs((ob.CoordinatesX(1)-ob.CoordinatesX(0)))<0.1)    Or    (Abs((ob.CoordinatesY(1)-
ob.CoordinatesY(0)))<0.1) Then
            EPC_SF_Path_OnHitTest=True
            Exit Function
        End If

        slope=(ob.CoordinatesY(1)-ob.CoordinatesY(0))/(ob.CoordinatesX(1)-ob.CoordinatesX(0))

         Temp1 = (x/slope+y-ob.CoordinatesY(0)+slope*ob.CoordinatesX(0))/(slope+1/slope)
         Temp2 = slope*(Temp1-ob.CoordinatesX(0))+ob.CoordinatesY(0)

        If   ( Sqr( (Temp1-x)^2)+  (Temp2-y)^2) < 20 Then
            EPC_SF_Path_OnHitTest=True
        End If

End Function

Public Sub EPC_SF_Path_OnMove(ob As CFCSim_ModelingElementInstance, ByVal x1 As Single, ByVal
y1 As Single, ByVal x2 As Single, ByVal y2 As Single)
          With ob
              If Sqr((.CoordinatesX(0) - x1) ^ 2 + (.CoordinatesY(0) - y1) ^ 2) <= 10 Then
               .CoordinatesX(0) = .CoordinatesX(0) + (x2 - x1)
               .CoordinatesY(0) = .CoordinatesY(0) + (y2 - y1)

                 .ConnectionPoints("c1").x = .ConnectionPoints("c1").x + (x2 - x1)
               .ConnectionPoints("c1").y = .ConnectionPoints("c1").y + (y2 - y1)

             ElseIf Sqr((.CoordinatesX(1) - x1) ^ 2 + (.CoordinatesY(1) - y1) ^ 2) <= 10 Then
               .CoordinatesX(1) = .CoordinatesX(1) + (x2 - x1)
               .CoordinatesY(1) = .CoordinatesY(1) + (y2 - y1)

                         .ConnectionPoints("c2").x = .ConnectionPoints("c2").x + (x2 - x1)
               .ConnectionPoints("c2").y = .ConnectionPoints("c2").y + (y2 - y1)

             Else
               .CoordinatesX(0) = .CoordinatesX(0) + (x2 - x1)
               .CoordinatesY(0) = .CoordinatesY(0) + (y2 - y1)
               .CoordinatesX(1) = .CoordinatesX(1) + (x2 - x1)
               .CoordinatesY(1) = .CoordinatesY(1) + (y2 - y1)

               .ConnectionPoints("c1").x = .ConnectionPoints("c1").x + (x2 - x1)
               .ConnectionPoints("c1").y = .ConnectionPoints("c1").y + (y2 - y1)
               .ConnectionPoints("c2").x = .ConnectionPoints("c2").x + (x2 - x1)
               .ConnectionPoints("c2").y = .ConnectionPoints("c2").y + (y2 - y1)
                 End If
          End With

End Sub
```

```
Public Function EPC_SF_Path_OnRelationValid(srcCP As CFCSim_ConnectionPoint, dstCP As
CFCSim_ConnectionPoint) As Boolean
        EPC_SF_Path_OnRelationValid=True

        If srcCP.RelationsTo.Count>0 Then
            MessagePrompt "Only one relation is allowed from this connection point "
            EPC_SF_Path_OnRelationValid=False
        End If

    End Function

    Public Sub EPC_SF_Path_OnSimulationInitialize(ob As CFCSim_ModelingElementInstance)
        ob.AddEvent "StartTravel",True
        ob.AddEvent "FinishTravel"
    End Sub

    Public Sub EPC_SF_Path_OnSimulationInitializeRun(ob As CFCSim_ModelingElementInstance,
RunNum As Integer)

End Sub

Public Function EPC_SF_Path_OnValidateParameters(ob As CFCSim_ModelingElementInstance,
Parameters As Object) As Boolean
        EPC_SF_Path_OnValidateParameters=ob.OnValidateParameters(Parameters,True)

        If EPC_SF_Path_OnValidateParameters=False Then Exit Function

End Function
```

DrawingTool - EPC_SF_Outline

```
Public Function EPC_SF_Outline_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y As
Single) As Boolean
        EPC_SF_Outline_OnCreate=True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+50
        ob.CoordinatesY(1)=y+50

End Function

Public Sub EPC_SF_Outline_OnDraw(ob As CFCSim_ModelingElementInstance)

        CDC.ChangeLineStyle CFC_DOT,1,RGB(230,230,230)
        CDC.MoveTo ob.CoordinatesX(0),ob.CoordinatesY(0)
        CDC.LineTo ob.CoordinatesX(1),ob.CoordinatesY(1)

End Sub

Public Sub EPC_SF_Outline_OnGetBoundingRect(ob As CFCSim_ModelingElementInstance, mRect As
CFCGraphics_Rect, HitTest As Boolean)
```

```
        Dim t As Single
        t=5

        If ob.CoordinatesX(0)<ob.CoordinatesX(1) Then
                mRect.Left=ob.CoordinatesX(0)-t
                mRect.Right=ob.CoordinatesX(1)+t
        Else
                mRect.Left=ob.CoordinatesX(1)-t
                mRect.Right=ob.CoordinatesX(0)+t
        End If

        If ob.CoordinatesY(0)<ob.CoordinatesY(1) Then
                mRect.top=ob.CoordinatesY(0)-t
                mRect.bottom=ob.CoordinatesY(1)+t
        Else
                mRect.top=ob.CoordinatesY(1)-t
                mRect.bottom=ob.CoordinatesY(0)+t
        End If

End Sub

Public Function EPC_SF_Outline_OnHitTest(ob As CFCSim_ModelingElementInstance, x As Single, y As
Single) As Boolean
        EPC_SF_Outline_OnHitTest=False
End Function
```

DrawingTool - EPC_SF_Label

```
Public Function EPC_SF_Label_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y As
Single) As Boolean

        EPC_SF_Label_OnCreate=True
        With ob
                .SetNumCoordinates 1
    .CoordinatesX(0) = x
    .CoordinatesY(0) = y
    .AddAttribute "Value","Value",CFC_Text,CFC_Single,CFC_Hidden
                ob("Value")="Test"
  End With

End Function

Public Sub EPC_SF_Label_OnDraw(ob As CFCSim_ModelingElementInstance)

        With ob
                CDC.TextOut  .CoordinatesX(0),.CoordinatesY(0),.Attr("Value")
        End With
End Sub

Public Sub EPC_SF_Label_OnGetBoundingRect(ob As CFCSim_ModelingElementInstance, mRect As
CFCGraphics_Rect, HitTest As Boolean)

        With ob
		mRect.Left=.CoordinatesX(0)
		mRect.top=.CoordinatesY(0)
		mRect.Right=mRect.Left+CDC.TextWidth(.Attr("Value"))
		mRect.bottom=mRect.top+CDC.TextHeight(.Attr("Value"))
	End With

End Sub

Public Function EPC_SF_Label_OnHitTest(ob As CFCSim_ModelingElementInstance, x As Single, y As
Single) As Boolean
	EPC_SF_Label_OnHitTest=False
End Function
```

4. MODULE ASSEMBLY TEPLATE

EPC_MA_SIM

```
Public Function EPC_MA_SIM_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y As
Single) As Boolean
	EPC_MA_SIM_OnCreate=True
	ob.OnCreate x,y,True

	ob.SetNumCoordinates 2
	ob.CoordinatesX(0)=x
	ob.CoordinatesY(0)=y
	ob.CoordinatesX(1)=x+200
	ob.CoordinatesY(1)=y+150

	ob.AddAttribute "ActualWorkingTime", "Actual Working Time (Hours/Day)", CFC_Numeric,
CFC_Single, CFC_ReadWrite,1
	ob.AddAttribute "FinishedDrawings", "Quantity of Spool Drawings Finished So Far",
CFC_Numeric, CFC_Single, CFC_ReadOnly
	ob.AddStatistic "DrawingCycleTime","Drawing Cycle Time",False,True
	ob.AddStatistic "Production","Production (Drawings/day)",False,True

	ob("ActualWorkingTime")=420

End Function

Public Sub EPC_MA_SIM_OnDraw(ob As CFCSim_ModelingElementInstance)
	CDC.RenderPicture "Module.jpg",ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1)-
ob.CoordinatesX(0),ob.CoordinatesY(1)-ob.CoordinatesY(0)
	CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)

	If ob.Selected Then
		CDC.ChangeLineStyle CFC_SOLID,1,RGB(255,0,0)
		CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
	End If
	ob.DrawConnectionPoints

End Sub
```

```
Public Sub EPC_MA_SIM_OnSimulationInitializeRun(ob As CFCSim_ModelingElementInstance, RunNum
As Integer)
        ob("FinishedDrawings")=0
End Sub
```

EPC_MA_1stLayer

```
Public Function EPC_MA_1stLayer_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y
As Single) As Boolean
        EPC_MA_1stLayer_OnCreate=True
        ob.OnCreate x,y,True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+300
        ob.CoordinatesY(1)=y+100

End Function

Public Sub EPC_MA_1stLayer_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.RenderPicture "1stLayer.bmp",ob.CoordinatesX(0)+1,ob.CoordinatesY(0)+1,64,64
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),65,65
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)

        CDC.ChangeFont "Arial",24,True,False,False,False
        CDC.TextOut ob.CoordinatesX(0)+110,ob.CoordinatesY(0)+50, "1st Layer"

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        ob.DrawConnectionPoints

End Sub
```

EPC_MA_2ndLayer

```
Public Function EPC_MA_2ndLayer_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y
As Single) As Boolean
        EPC_MA_2ndLayer_OnCreate=True
        ob.OnCreate x,y,True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+300
        ob.CoordinatesY(1)=y+100

End Function

Public Sub EPC_MA_2ndLayer_OnDraw(ob As CFCSim_ModelingElementInstance)
```

```
        CDC.RenderPicture "2ndLayer.bmp",ob.CoordinatesX(0)+1,ob.CoordinatesY(0)+1,64,64
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),65,65
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)

        CDC.ChangeFont "Arial",24,True,False,False,False
        CDC.TextOut ob.CoordinatesX(0)+110,ob.CoordinatesY(0)+50, "2nd Layer"

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        ob.DrawConnectionPoints

End Sub
```

EPC_MA_3rdLayer

```
Public Function EPC_MA_3rdLayer_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y
As Single) As Boolean
        EPC_MA_3rdLayer_OnCreate=True
        ob.OnCreate x,y,True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+300
        ob.CoordinatesY(1)=y+100

End Function

Public Sub EPC_MA_3rdLayer_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.RenderPicture "3rdLayer.bmp",ob.CoordinatesX(0)+1,ob.CoordinatesY(0)+1,64,64
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),65,65
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)

        CDC.ChangeFont "Arial",24,True,False,False,False
        CDC.TextOut ob.CoordinatesX(0)+110,ob.CoordinatesY(0)+50, "3rd Layer"

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        ob.DrawConnectionPoints

End Sub
```

EPC_MA_RemainingWork

```
Public Function EPC_MA_RemainingWork_OnCreate(ob As CFCSim_ModelingElementInstance, x As
Single, y As Single) As Boolean
        EPC_MA_RemainingWork_OnCreate=True
        ob.OnCreate x,y,True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+80
        ob.CoordinatesY(1)=y+300

End Function

Public Sub EPC_MA_RemainingWork_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.RenderPicture "RemainingWork.bmp",ob.CoordinatesX(0)+1,ob.CoordinatesY(0)+1,64,64
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),65,65
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)

        CDC.ChangeFont "Arial",18,True,False,False,False
        CDC.TextOut ob.CoordinatesX(0)+2,ob.CoordinatesY(0)+120, "Remaining"
        CDC.TextOut ob.CoordinatesX(0)+20,ob.CoordinatesY(0)+140, "Work"

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        ob.DrawConnectionPoints

End Sub
```

5. SITE INSTALLATION TEMPLATE

EPC_SI_SIM

```
Public Function EPC_SI_SIM_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y As
Single) As Boolean
        EPC_SI_SIM_OnCreate=True
        ob.OnCreate x,y,True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+150
        ob.CoordinatesY(1)=y+200

        ob.AddAttribute "ActualWorkingTime", "Actual Working Time (Hours/Day )", CFC_Numeric,
CFC_Single, CFC_ReadWrite,1
        ob.AddAttribute "FinishedDrawings", "Quantity of Spool Drawings Finished So Far",
CFC_Numeric, CFC_Single, CFC_ReadOnly
        ob.AddStatistic "DrawingCycleTime","Drawing Cycle Time",False,True
        ob.AddStatistic "Production","Production (Drawings/day)",False,True

        ob("ActualWorkingTime")=420

End Function
```

```
Public Sub EPC_SI_SIM_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.RenderPicture "Site.jpg",ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1)-
ob.CoordinatesX(0),ob.CoordinatesY(1)-ob.CoordinatesY(0)
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_SOLID,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        ob.DrawConnectionPoints

End Sub

Public Sub EPC_SI_SIM_OnSimulationInitializeRun(ob As CFCSim_ModelingElementInstance, RunNum
As Integer)
        ob("FinishedDrawings")=0
End Sub
```

6. PUBLIC MODULE ELEMENTS TEMPLATE

DatabaseImporter

```
Public Function DatabaseImporter_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y As
Single) As Boolean
        ob.OnCreate x,y,True

        DatabaseImporter_OnCreate=True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+60
        ob.CoordinatesY(1)=y+60

        ob.AddAttribute "Name","Product Name",CFC_Text, CFC_Single, CFC_ReadWrite
   ob.AddAttribute "Database","Database Source",CFC_Text,CFC_Single,CFC_ReadWrite
        ob.AddAttribute "Product","Table/Query for Product Definition",CFC_Text,
CFC_Single,CFC_ReadWrite

        ob("Name")="Product"
        ob("Database")="D:\RA_KBR\8.Large-scale Simulation\2.Drafting\Simulation.mdb"
        ob("Product")="ProductImport"

        ob.AddAttribute "NumAttribute","Number of Attributes",CFC_Numeric, CFC_Single,
CFC_ReadOnly
   ob.AddAttribute "NumProduct","Number of Products",CFC_Numeric, CFC_Single, CFC_ReadOnly
        ob.AddAttribute "ProductAttr", "Prodcut Attributes",CFC_Array, CFC_Table, CFC_ReadOnly

        ob.AddAttribute "Fired","Entites Created so far",CFC_Numeric, CFC_Single,CFC_Hidden

        ob.AddConnectionPoint "Out",x+65,y+30,COutput,5
End Function
```

```
Public Sub DatabaseImporter_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub DatabaseImporter_OnDraw(ob As CFCSim_ModelingElementInstance)
        'CDC.ChangeFont "Courier New",13,True,False,False,False

        CDC.Circ ob.CoordinatesX(0)+30,ob.CoordinatesY(0)+30,30
        CDC.RenderPicture
"DatabaseImporter.bmp",ob.CoordinatesX(0)+10,ob.CoordinatesY(0)+15,32,32

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)+3,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)

        ob.DrawConnectionPoints
End Sub

Public Sub DatabaseImporter_OnSimulationInitialize(ob As CFCSim_ModelingElementInstance)
        ob.AddEvent "FireEntity",True

'**********IMPORT OUTSIDE DATABASE RECORDS TO INNER ARRAY OF SIMULATION
MODEL***********************
        Dim myDB As Database
   Dim myRS As Recordset
   Dim numAttr As Integer
   Dim i, j As Integer
   j=0

'Open Outside Database
        Set myDB = OpenDatabase(ob!Database)
   Set myRS = myDB.OpenRecordset(ob!Product, dbOpenDynaset)

'Define Size of Array Inside Simphony Based On Size of Outside Database
        myRS.MoveLast              'Have to move to last one in order to count all records.
        ob("NumProduct")=CInt(myRS.RecordCount)      'Count Records (Row)
        ob("NumAttribute")=CInt(myRS.Fields.Count)   'Count Fields (Column)
        ob("ProductAttr").SetRC(ob("NumProduct"),ob("NumAttribute")) 'Define size of Array
(Row,Column)

'Read Attribute Names
        For i=0 To ob("NumAttribute")-1 Step 1
                ob("ProductAttr").ColumnLabel(i+1-1)= CStr(myRS.Fields(i+1-1).Name)
        Next i

'Read Attribute Values
        myRS.MoveFirst
        With ob("ProductAttr")
                Do While Not myRS.EOF
                        For i=0 To CInt(ob("NumAttribute")-1)
                                If IsNull(myRS.Fields(i).Value) Or IsEmpty(myRS.Fields(i).Value)
Then
```

```
                    .ValueRC(j-1+1,i-1+1)=""
                Else
                    .ValueRC(j-1+1,i-1+1)=myRS.Fields(i).Value
                End If
            Next i
            myRS.MoveNext 'Move to next row
            j=j+1
        Loop
    End With
'**********IMPORT OUTSIDE DATABASE RECORDS TO INNER ARRAY OF SIMULATION
MODEL**********************

End Sub

Public Sub DatabaseImporter_OnSimulationInitializeRun(ob As CFCSim_ModelingElementInstance,
RunNum As Integer)
    ob("Fired")=0
    ob.ScheduleEvent ob.AddEntity,"FireEntity",ob("ProductAttr").ValueRC(0,ob("NumAttribute")-1)

End Sub

Public Sub DatabaseImporter_OnSimulationProcessEvent(ob As CFCSim_ModelingElementInstance,
MyEvent As String, Entity As CFCSim_Entity)

'**********GIVE THE VALUE OF ARRAY TO ENTITY ROW BY ROW**********************
    Dim AttributeName As String
    Dim newEntity As CFCSim_Entity
    Dim i As Integer

    If ob("Fired")>= ob("NumProduct") Then Exit Sub

    Set newEntity = ob.AddEntity
    For i= 0 To ob("NumAttribute")-1
        AttributeName=ob("ProductAttr").ColumnLabel(i-1+1)
        newEntity(AttributeName)=ob("ProductAttr").ValueRC(ob("fired")-1+1,i-1+1)
    Next i

    ob.TransferOut newEntity, ob.ConnectionPoints("Output")

    ob("fired")=ob("fired")+1
    If ob("Fired")< ob("NumProduct") Then
        ob.ScheduleEvent Entity, "FireEntity", ob("ProductAttr").ValueRC(ob("fired"),
ob("NumAttribute")-1)-SimTime
    End If
'**********GIVE THE VALUE OF ARRAY TO ENTITY ROW BY ROW**********************

End Sub
```

TimeCollector

```
Option Explicit

Public Function TimeCollector_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y As
Single) As Boolean
```

```
        ob.OnCreate x,y,True
        TimeCollector_OnCreate=True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+50
        ob.CoordinatesY(1)=y+50

        ob.AddAttribute "ProductID", "Product ID (An Passing Entity's Attribute Name Indicating Its
ID)", CFC_Text, CFC_Single, CFC_ReadWrite
        ob.AddAttribute "TimeType", "Time Type (In/Out) To Collect In A Location",
CFC_Text,CFC_ListBox, CFC_ReadWrite
        ob("ProductID")="ProductID"
        ob("TimeType")="In"

        ob.AddAttribute "DatabaseExporter","",CFC_Numeric, CFC_Single, CFC_Hidden 'record the
DatabaseExporter element id

        ob.AddConnectionPoint "In" , x, y+25, CInput, 5
        ob.AddConnectionPoint "Out",x+50,y+25,COutput,5

End Function

Public Sub TimeCollector_OnListBoxInitialize(ob As CFCSim_ModelingElementInstance, attr As
CFCSim_Attribute, lstList As Object)
        lstList.additem "In"
        lstList.additem "Out"
End Sub

Public Sub TimeCollector_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub TimeCollector_OnDraw(ob As CFCSim_ModelingElementInstance)
'       CDC.TextOut ob.CoordinatesX(0)+15,ob.CoordinatesY(0)+20, "Time"
'       CDC.TextOut ob.CoordinatesX(0)+5,ob.CoordinatesY(0)+30, "Collector"
        CDC.RenderPicture "TimeCollector.bmp",ob.CoordinatesX(0),ob.CoordinatesY(0),50,50

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)+3,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)

        ob.DrawConnectionPoints
End Sub

Public Sub TimeCollector_OnSimulationInitialize(ob As CFCSim_ModelingElementInstance)
        Dim DatabaseExporter As CFCSim_ModelingElementInstance
                Dim i As Long

        'Locate DatabaseExporter element
        For Each DatabaseExporter In Elements
                        Tracer.Trace "Checked " & i & " elements"    'OK!
        If DatabaseExporter.ElementType="DatabaseExporter" Then
            ob("DatabaseExporter")=CStr(DatabaseExporter.Id)
            Tracer.Trace "Found DatabaseExporter"     'OK!

            Exit For
        End If
        i=i+1
    Next

End Sub

Public Sub TimeCollector_OnSimulationTransferIn(ob As CFCSim_ModelingElementInstance, Entity As
CFCSim_Entity, SrcCp As CFCSim_ConnectionPoint, DstCp As CFCSim_ConnectionPoint)
    Dim DatabaseExporter As CFCSim_ModelingElementInstance
    Dim batchEnt As CFCSim_Entity
    Dim batchEle As CFCSim_ModelingElementInstance

    Set DatabaseExporter=Elements(CStr(ob("DatabaseExporter")))

    Select Case ob("TimeType")

    '**********COLLECT TIME OF ENTERING A LOCATION***********************
        Case "In"

            '******If this is a normal entity******
            If Entity("batchLoc")="" Then
                With DatabaseExporter("Output")

    .ValueRC(DatabaseExporter("OPIndex"),0)=Entity(CStr(ob("ProductID"))) 'Product id

    .ValueRC(DatabaseExporter("OPIndex"),1)=ob.Parent("Description") 'Location Name
                    .ValueRC(DatabaseExporter("OPIndex"),2)=SimTime
'Entering time
                End With
                Entity("EOPIndex")=DatabaseExporter("OPIndex")'This save the
index of this entity's record
                DatabaseExporter("OPIndex")=DatabaseExporter("OPIndex")+1
    'Next record in the output array

            '******If this is a batched entity******
            Else
                Set batchEle=Elements(CStr(Entity("batchLoc")))
                With batchEle.File("batchQueue") 'Unbatch the batch entity
                    .MoveFirst 'Start from top of the batch queue
                    While (.EOF=False)        'Look for the arrived batch in the
whole batch queue
                        If .entity.Id=Entity("BatchFirstID") Then 'Found the
arrived batch in the whole batch queue

                            '***START COLLECTING TIME FOR

EACH ENTITY OF THIS BATCH***
                            While (.EOF=False And .Length>0)
                                Set batchEnt=.entity
                                With DatabaseExporter("Output")
    .ValueRC(DatabaseExporter("OPIndex"),0)=batchEnt("id") 'Product id

    .ValueRC(DatabaseExporter("OPIndex"),1)=ob.Parent("Description") 'Location Name

    .ValueRC(DatabaseExporter("OPIndex"),2)=SimTime 'Entering time
                                End With

    .entity("EOPIndex")=DatabaseExporter("OPIndex")'This save the index of this entity's record

    DatabaseExporter("OPIndex")=DatabaseExporter("OPIndex")+1        'Next record in the output
array

                                'Check if all entities in this batch
have been added In time

                                If
batchEnt.Id=Entity("BatchLastID") Then
                                    ob.TransferOut Entity
                                    Exit Sub
                                End If

                                .MoveNext
                            Wend
                            '***FINISH COLLECTING TIME FOR
EACH ENTITY OF THIS BATCH***

                        Else
                            .MoveNext
                        End If
                    Wend    'Look for the arrived batch in the whole batch queue
                End With
            End If

    '***********COLLECT TIME OF LEAVING A LOCATION***********************
        Case "Out"

            '******If this is a normal entity******
            If Entity("batchLoc")="" Then
                DatabaseExporter("Output").ValueRC(Entity("EOPIndex"),
3)=SimTime 'Leaving time

            '******If this is a batched entity******
            Else
                Set batchEle=Elements(CStr(Entity("batchLoc")))
                With batchEle.File("batchQueue") 'Unbatch the batch entity
                    .MoveFirst 'Start from top of the batch queue
                    While (.EOF=False)          'Look for the arrived batch in the
whole batch queue
                        If .entity.Id=Entity("BatchFirstID") Then 'Found the
arrived batch in the whole batch queue

                            '***START COLLECTING TIME FOR
EACH ENTITY OF THIS BATCH***
                            While (.EOF=False And .Length>0)
                                Set batchEnt=.entity
    DatabaseExporter("Output").ValueRC(batchEnt("EOPIndex"),3)=SimTime 'Leaving time
                                'Check if all entities in this batch
have been added Out time
                                If
batchEnt.Id=Entity("BatchLastID") Then
                                    ob.TransferOut Entity
                                    Exit Sub
                                End If

                                .MoveNext
                            Wend
                            '***FINISH COLLECTING TIME FOR
EACH ENTITY OF THIS BATCH***

                        Else
                            .MoveNext
                        End If
                    Wend    'Continue looking for the arrived batch in the whole
batch queue
                End With
            End If

    End Select
    ob.TransferOut Entity

End Sub
```

DatabaseExporter

```
Public Function DatabaseExporter_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y As
Single) As Boolean
    ob.OnCreate x,y,True

    DatabaseExporter_OnCreate=True

    ob.SetNumCoordinates 2
    ob.CoordinatesX(0)=x
    ob.CoordinatesY(0)=y
    ob.CoordinatesX(1)=x+60
    ob.CoordinatesY(1)=y+60

    ob.AddAttribute "Database","Database Source",CFC_Text,CFC_Single,CFC_ReadWrite
    ob("Database")="D:\RA_KBR\8.Large-scale Simulation\2.Drafting\Simulation.mdb"

    ob.AddAttribute "Output", "Output Array",CFC_Array, CFC_Table, CFC_ReadOnly
    ob.AddAttribute "OPIndex","Output Index",CFC_Numeric, CFC_Single, CFC_Hidden

    ob.AddConnectionPoint "In",x-3,y+30,COutput,5
End Function
Public Sub DatabaseExporter_OnDragDraw(ob As CFCSim_ModelingElementInstance)
    ob.OnDraw
End Sub
```

```
Public Sub DatabaseExporter_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.ChangeFont "Courier New",13,True,False,False,False

        CDC.Circ ob.CoordinatesX(0)+30,ob.CoordinatesY(0)+30,30
        CDC.RenderPicture
"DatabaseExporter.bmp",ob.CoordinatesX(0)+10,ob.CoordinatesY(0)+15,32,32

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)+3,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)

        ob.DrawConnectionPoints
End Sub

Public Sub DatabaseExporter_OnSimulationInitialize(ob As CFCSim_ModelingElementInstance)
        ob("Output").SetRC(30000,4) 'Limited space to store time record for entity
End Sub

Public Sub DatabaseExporter_OnSimulationInitializeRun(ob As CFCSim_ModelingElementInstance,
RunNum As Integer)
        ob("OPIndex")=0
End Sub

Public Sub DatabaseExporter_OnSimulationPostRun(ob As CFCSim_ModelingElementInstance, RunNum
As Integer)
        Dim sql As String
        Dim id As Long
        Dim j As Long
        Dim k As Integer
        Dim m As Integer
        Dim myDB As Database
  Dim myRS As Recordset

'**********Setup Database Connection*****************
        Set myDB = OpenDatabase(ob!Database)

        'Clean the old record of output table of database
        If RunNum=1 Then
                sql = "Delete * From Output"
                myDB.Execute sql
        End If

        'From 2nd run, need to put the point at a new record in order to keep all records of previous run,
        'because id is redefined and becomes 0 again after every run.
        sql="Select Max(id) as MaxID from output"
        Set myRS = myDB.OpenRecordset(sql, dbOpenDynaset)
        If myRS!MaxID="" Or IsNull(myRS!MaxID) Then
                id=1
        Else
                id=myRS!MaxID+1
        End If
        'Open the output table of database
        sql = "Select * From Output"
    Set myRS = myDB.OpenRecordset(sql, dbOpenDynaset, dbAppendOnly)

'**********EXPORT INNER ARRAY OF SIMULATION MODEL TO OUTSIDE DATABASE**********
        For j=0 To 30000-1
                If ob("Output").ValueRC(j-1+1,0)="" Then Exit Sub 'If there is no record in this row,
just Exit

                myRS.AddNew
            myRS!id = id
            myRS!Product_id = ob("Output").ValueRC(j-1+1,0) 'Product_ID
                myRS!LocName = ob("Output").ValueRC(j-1+1,1) 'Location
                myRS!InTime = ob("Output").ValueRC(j-1+1,2) 'Enter time
                If ob("Output").ValueRC(j-1+1,3)<>"" Then myRS!OutTime = ob("Output").ValueRC(j-
1+1,3) 'Leaving time
                myRS!Run=RunNum
                myRS.Update
            id=id+1

                'Delete the values of this record in the array
            For k=0 To 3
                        ob("Output").ValueRC(j-1+1,k)=""
                Next k

        Next j
'**********EXPORT INNER ARRAY OF SIMULATION MODEL TO OUTSIDE DATABASE**********

End Sub
```

Batch

```
Dim FirstEntity As CFCSim_Entity   'Create a new entity as the group of batched entities

Public Function Batch_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y As Single) As
Boolean
        ob.OnCreate x,y,True
        Batch_OnCreate=True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+50
        ob.CoordinatesY(1)=y+50

        ob.AddAttribute "BatchThre","Batch Threshold",CFC_Numeric, CFC_Single, CFC_ReadWrite
        ob.AddAttribute "MaxWaitTime","Maximum Waiting Time (Minute)",CFC_Numeric, CFC_Single,
CFC_ReadWrite

        ob.AddAttribute "Count","Number of entities of the batch",CFC_Single, CFC_Single,
CFC_Hidden
        ob.AddAttribute "BatchFirstID","ID of the first entity of the batch",CFC_Single, CFC_Single,
CFC_Hidden
```

```
        ob.AddFile "BatchQueue",QUEUE   'Create A Batch Queue

        ob("BatchThre")=1
        ob("MaxWaitTime")=60
        ob("TimeStep")=10
        ob("Count")=0

        ob.AddConnectionPoint "In" , x-10, y+25, CInput,5
        ob.AddConnectionPoint "Out",x+60,y+25,COutput,5

End Function

Public Sub Batch_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub Batch_OnDraw(ob As CFCSim_ModelingElementInstance)

        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)
        CDC.MoveTo ob.CoordinatesX(0), ob.CoordinatesY(0)
        CDC.LineTo ob.CoordinatesX(0), ob.CoordinatesY(0)+50
        CDC.LineTo ob.CoordinatesX(0)+50, ob.CoordinatesY(0)+25
        CDC.LineTo ob.CoordinatesX(0), ob.CoordinatesY(0)

        CDC.ChangeFont "Arial",12,True,False,False,False
        CDC.TextOut ob.CoordinatesX(0)+5,ob.CoordinatesY(0)+20,"Batch"

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)

        ob.DrawConnectionPoints
End Sub

Public Sub Batch_OnSimulationInitialize(ob As CFCSim_ModelingElementInstance)
        ob.AddEvent "Fire",True
End Sub

Public Sub Batch_OnSimulationInitializeRun(ob As CFCSim_ModelingElementInstance, RunNum As
Integer)
        ob("Count")=0
End Sub

Public Sub Batch_OnSimulationProcessEvent(ob As CFCSim_ModelingElementInstance, MyEvent As
String, Entity As CFCSim_Entity)
        'If the entities have waited time longer than MaxWaitTime, but haven't formed a batch to be
released, these entities should be released as a batch.
        Dim NewEntity As CFCSim_Entity   'Create a new entity as the group for entities which has
waited too long time

        Set NewEntity=ob.CloneEntity(Entity)
        NewEntity("BatchFirstID")=ob("BatchFirstID") 'Record the last entity ID in this batch
        ob.File("BatchQueue").MoveLast
        NewEntity("BatchLastID")=ob.File("BatchQueue").entity.Id 'Record the last entity ID in this
batch??????????????
        NewEntity("BatchLocID")=ob.Id 'Record the location where entities are batched

        ob.TransferOut NewEntity, ob.ConnectionPoints("Out") 'Fire a new entity
        ob("Count")=0 'Reset number of entities of the batch

End Sub

Public Sub Batch_OnSimulationTransferIn(ob As CFCSim_ModelingElementInstance, Entity As
CFCSim_Entity, SrcCp As CFCSim_ConnectionPoint, DstCp As CFCSim_ConnectionPoint)

'*****************BATCH FUNCTIONS*******************************************************
        Dim NewEntity As CFCSim_Entity   'Create a new entity as the group of batched entities

'Add entity to the BatchQueue
        ob.File("BatchQueue").Add Entity,0 'No priority used**********

        If ob("Count")=0 Then
                ob("BatchFirstID")=Entity.Id 'Record the first entity ID in this batch
                Set FirstEntity=Entity
                ob.ScheduleEvent FirstEntity,"Fire",ob("MaxWaitTime")
        End If
        ob("Count")=ob("Count")+1 'Update number of entities of the batch
        Tracer.Trace " Count=" & ob("Count") & " Threshold=" & ob("BatchThre")

'Check if the batch threshold value is reached or not
        If ob("Count")>=ob("BatchThre") Then
                Set NewEntity=ob.CloneEntity(Entity)
                NewEntity("BatchFirstID")=ob("BatchFirstID") 'Record the last entity ID in this batch
                NewEntity("BatchLastID")=Entity.Id 'Record the last entity ID in this batch
                NewEntity("BatchLocID")=ob.Id 'Record the location where entities are batched

                ob.TransferOut NewEntity, ob.ConnectionPoints("Out") 'Fire a new entity
                ob("Count")=0 'Reset number of entities of the batch

                ob.CancelEvent FirstEntity,"Fire"
        Else
                Exit Sub
        End If
'*****************BATCH FUNCTIONS*******************************************************

End Sub
```

Unbatch

```
Public Function Unbatch_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y As Single)
As Boolean
        ob.OnCreate x,y,True
        Unbatch_OnCreate=True

        ob.SetNumCoordinates 2
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+50
        ob.CoordinatesY(1)=y+50

        ob.AddConnectionPoint "In" , x-10, y+25, CInput, 5
        ob.AddConnectionPoint "Out",x+60,y+25,COutput,5

End Function

Public Sub Unbatch_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub Unbatch_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)
        CDC.MoveTo ob.CoordinatesX(0), ob.CoordinatesY(0)+25
        CDC.LineTo ob.CoordinatesX(0)+50, ob.CoordinatesY(0)+50
        CDC.LineTo ob.CoordinatesX(0)+50, ob.CoordinatesY(0)
        CDC.LineTo ob.CoordinatesX(0), ob.CoordinatesY(0)+25

        CDC.ChangeFont "Arial",12,True,False,False,False
        CDC.TextOut ob.CoordinatesX(0)+10,ob.CoordinatesY(0)+20,"Unbatch"

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)

        ob.DrawConnectionPoints
End Sub

Public Sub Unbatch_OnSimulationTransferIn(ob As CFCSim_ModelingElementInstance, Entity As
CFCSim_Entity, SrcCp As CFCSim_ConnectionPoint, DstCp As CFCSim_ConnectionPoint)

'****************UNBATCH FUNCTIONS*************************************************
        Dim BatchEle As CFCSim_ModelingElementInstance
        Dim BatchEnt As CFCSim_Entity

        Set BatchEle=Elements(CStr(Entity("BatchLocID"))) 'Find out which element the batch was made
at
        If BatchEle.File("BatchQueue").Length> 0 Then
                With BatchEle.File("BatchQueue")
                        .MoveFirst 'Start checking from top of the whole batch queue

                        While (.EOF=False)          'Look for the arrived batch in the whole batch queue
                                If .entity.Id=Entity("BatchFirstID") Then 'Found the arrived batch in
the whole batch queue
                                        '***START UNBATCHING***
                                        While (.EOF=False And .Length>0)
                                                Set BatchEnt=.entity
                                                ob.TransferOut BatchEnt 'Release the original entity
one by one

                                                'Check if all entities in this batch have been
unbatched
```

```
                        If BatchEnt.Id=Entity("BatchLastID") Then
                            .Remove BatchEnt 'Delete the entity from
the batch queue
                            ob.DeleteEntity Entity 'Delete the batch
entity
                            Exit Sub
                        End If

                        .MoveNext
                        .Remove BatchEnt 'Delete the entity from the batch
queue
                    Wend
                    '***FINISH UNBATCHING***
                Else
                    .MoveNext
                End If
            Wend                                'Look for the arrived batch in the whole
batch queue

        End With
    Else
        MessagePrompt "No entity has been batched!"
    End If
'****************UNBATCH FUNCTIONS**************************************************
End Sub
```

Assembly

```
Public Function Assembly_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y As Single)
As Boolean
    ob.OnCreate x,y,True
    Assembly_OnCreate=True
    ob.AddAttribute "AssemblyCriteria","Criteria to Assembly (An Attribute of Entities)",CFC_Text,
CFC_Single, CFC_ReadWrite
    ob.AddAttribute "Quantity1","Assembly Threshold (An Attribute Name If Based On Entity
Attribute)",CFC_Text, CFC_Single, CFC_ReadWrite
    ob.AddAttribute "Quantity2","Assembly Threshold (A Constant Value If It Is
Constant)",CFC_Numeric,CFC_Single, CFC_ReadWrite

    ob.AddFile "Parts_Queue",QUEUE
    ob.AddAttribute "Products","Collection of the
products",CFC_Collection,CFC_Single,CFC_Hidden

    ob.SetNumCoordinates 2
    ob.CoordinatesX(0)=x
    ob.CoordinatesY(0)=y
    ob.CoordinatesX(1)=x+60
    ob.CoordinatesY(1)=y+60

    ob("AssemblyCriteria")="ProductID"
    ob("Quantity1")=""
    ob("Quantity2")=CInt(0)

    ob.AddConnectionPoint "In" , x-10, y+30, CInput, 5
```

```
        ob.AddConnectionPoint "Out",x+70,y+30,COutput,5

End Function

Public Sub Assembly_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub Assembly_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.ChangeFont "Courier New",13,True,False,False,False

        CDC.RenderPicture "Assembly.bmp",ob.CoordinatesX(0)+1,ob.CoordinatesY(0)+1,32,32
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)
        CDC.TextOut ob.CoordinatesX(0)+5,ob.CoordinatesY(0)+40, "Assembly"

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)

        ob.DrawConnectionPoints

        CDC.ChangeFont "Arial",11,True,False,False,False

End Sub

Public Sub Assembly_OnSimulationInitializeRun(ob As CFCSim_ModelingElementInstance, RunNum As
Integer)
        ob.File("Parts_Queue").Clear
        ob("Products").Collection.Clear
End Sub

Public Sub Assembly_OnSimulationTransferIn(ob As CFCSim_ModelingElementInstance, Entity As
CFCSim_Entity, SrcCp As CFCSim_ConnectionPoint, DstCp As CFCSim_ConnectionPoint)
        Dim newEntity As CFCSim_Entity

        Dim ProductId As Variant
        Dim Existed As Boolean
        Dim Quantity As Integer
        Dim Count As Integer
        Dim ProEnt As CFCSim_Entity
        Dim RemEnt As CFCSim_Entity

        Existed=False

        ob.File("Parts_Queue").Add Entity,0

        '***********************************************************************************
********************
        'Check if the part's ProductID is in the product collection, which lists all ProductID of parts
        'waiting in the Parts_Queue.
        'If No, add the ProductID in the collection.
        '***********************************************************************************
********************
```

```
        For Each ProductId In ob("Products").Collection
                If Entity(CStr(ob("AssemblyCriteria")))=ProductId Then
                        Existed=True
                        Exit For
                End If
        Next

        If Existed=False Then
                ob("Products").Collection.Add
Entity(CStr(ob("AssemblyCriteria"))),Entity(CStr(ob("AssemblyCriteria")))
        End If


        '************************************************************************************
********************
        'Check ProductID one by one in the Products Collection.
        'For each ProductID:
        'Count number of parts of this Product in the Parts_Queue to compare with the actual total
number of parts of this Product.
        'If unequal, End sub
        'If equual, 1.Remove all parts of this Product one by one from Parts_Queue;
        '                       2.Remove this ProductID from the Collection;
        '                       3.Release one entity representing the Product.
        '************************************************************************************
*******************

        'Check ProductID one by one in the product collection
        For Each ProductId In ob("Products").Collection

                'Count number of parts of the Product, which the newly arrived part belongs to, in the
Parts_Queue
                Count=0
                With ob.File("Parts_Queue")
                        If .Length<>0 Then
                                .MoveFirst
                                While (.EOF=False And .Length>0)
                                        If .entity(CStr(ob("AssemblyCriteria")))=ProductId Then
                                                Count=Count+1
                                                Set ProEnt=.entity
                                        End If
                                        .MoveNext
                                Wend
                        End If
                End With

                'Get actual total number of parts of this Product
                If ob("Quantity2")=0 Then
                        Quantity=ProEnt(CStr(ob("Quantity1")))
                Else
                        Quantity=ob("Quantity2")
                End If

                'If number of parts of this Product in the Parts_Queue is equual to
                'the actual total number of parts of this Product.
                If Quantity=Count Then
```

```
            '1.Remove All parts of this Product one by one from Parts_Queue;
            With ob.File("Parts_Queue")
                If .Length<>0 Then
                    .MoveFirst
                    While (.EOF=False And .Length>0)
                        If .entity(CStr(ob("AssemblyCriteria")))=ProductId
Then
                            Set RemEnt =.entity
                            .Remove RemEnt
                        End If
                        If (.EOF=False And .Length>0) Then
                            .MoveNext
                        End If
                    Wend
                End If
            End With

            '2.Remove this ProductID from the Collection;
            ob("Products").Collection.Remove CStr(ProductId)

            '3.Release one entity representing the Product.
            Set newEntity=ob.CloneEntity(ProEnt)
            ob.TransferOut newEntity, ob.ConnectionPoints("Out")

            Exit For        'This function is triggered once an entity enters in "Assembly"
element. It can only cause
                        'one product to be assemblied. So it can stop
searching the remaining Products by "Exit For".
        End If
    Next

End Sub
```

SCMatching

```
Public Function SCMatch_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y As Single)
As Boolean
    ob.OnCreate x,y,True
    SCMatch_OnCreate=True
    ob.AddAttribute "MatchCriteria","Criteria to Match (An Attribute of Entities)",CFC_Text,
CFC_Single, CFC_ReadWrite
    'ob.AddAttribute "Quantity1","Assembly Threshold (An Attribute Name If Based On Entity
Attribute)",CFC_Text, CFC_Single, CFC_ReadWrite
    ob.AddAttribute "Quantity","Match Threshold (Number of Entities to Match
Here)",CFC_Numeric,CFC_Single, CFC_ReadWrite

    ob.AddFile "Parts_Queue",QUEUE
    ob.AddAttribute "Products","Collection of the
products",CFC_Collection,CFC_Single,CFC_Hidden

    ob.SetNumCoordinates 2
    ob.CoordinatesX(0)=x
    ob.CoordinatesY(0)=y
    ob.CoordinatesX(1)=x+70
```

```
        ob.CoordinatesY(1)=y+60

        ob("MatchCriteria")="ProductID"
        'ob("Quantity1")=""
        ob("Quantity")=CInt(0)

        ob.AddConnectionPoint "In" , x-5, y+30, CInput, 5
        ob.AddConnectionPoint "Out",x+75,y+30,COutput,5

End Function

Public Sub SCMatch_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub SCMatch_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.ChangeFont "Courier New",13,True,False,False,False

        CDC.RenderPicture "Match.bmp",ob.CoordinatesX(0)+1,ob.CoordinatesY(0)+1,32,32
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)
        CDC.TextOut ob.CoordinatesX(0)+2,ob.CoordinatesY(0)+32, "SupplyChain"
        CDC.TextOut ob.CoordinatesX(0)+2,ob.CoordinatesY(0)+45, "Matching"

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)

        ob.DrawConnectionPoints

        CDC.ChangeFont "Arial",11,True,False,False,False

End Sub

Public Sub SCMatch_OnSimulationTransferIn(ob As CFCSim_ModelingElementInstance, Entity As
CFCSim_Entity, SrcCp As CFCSim_ConnectionPoint, DstCp As CFCSim_ConnectionPoint)
        Dim newEntity As CFCSim_Entity

        Dim ProductId As Variant
        Dim Existed As Boolean
        Dim Quantity As Integer
        Dim Count As Integer
        Dim ProEnt As CFCSim_Entity
        Dim RemEnt As CFCSim_Entity

        Existed=False

        ob.File("Parts_Queue").Add Entity,0

        '**********************************************************************************************
*********************
        'Check if the part's ProductID is in the product collection, which lists all ProductID of parts
        'waiting in the Parts_Queue.
        'If No, add the ProductID in the collection.
        '*******************************************************************************
*******************
        For Each ProductId In ob("Products").Collection
                If Entity(CStr(ob("MatchCriteria")))=ProductId Then
                        Existed=True
                        Exit For
                End If
        Next

        If Existed=False Then
                ob("Products").Collection.Add
Entity(CStr(ob("MatchCriteria"))),Entity(CStr(ob("MatchCriteria")))
        End If


        '*******************************************************************************
*******************
        'Check ProductID one by one in the Products Collection.
        'For each ProductID:
        'Count number of parts of this Product in the Parts_Queue to compare with the actual total
number of parts of this Product.
        'If unequal, End sub
        'If equual, 1.Remove all parts of this Product one by one from Parts_Queue;
        '                       2.Remove this ProductID from the Collection;
        '                       3.Release one entity representing the Product.
        '*******************************************************************************
*******************

        'Check ProductID one by one in the product collection
        For Each ProductId In ob("Products").Collection

                'Count number of parts of the Product, which the newly arrived part belongs to, in the
Parts_Queue
                Count=0
                With ob.File("Parts_Queue")
                        If .Length<>0 Then
                                .MoveFirst
                                While (.EOF=False And .Length>0)
                                        If .entity(CStr(ob("MatchCriteria")))=ProductId Then
                                                Count=Count+1
                                                Set ProEnt=.entity
                                        End If
                                        .MoveNext
                                Wend
                        End If
                End With

                'Get actual total number of parts of this Product
                        'If ob("Quantity2")=0 Then
                        '       Quantity=ProEnt(CStr(ob("Quantity1")))
                        'Else
                Quantity=ob("Quantity")
                        'End If

                'If number of parts of this Product in the Parts_Queue is equual to
                'the actual total number of parts of this Product.
        If Quantity=Count Then

            '1.Remove All parts of this Product one by one from Parts_Queue;
            With ob.File("Parts_Queue")
                If .Length<>0 Then
                    .MoveFirst
                    While (.EOF=False And .Length>0)
                        If .entity(CStr(ob("MatchCriteria")))=ProductId
Then
                            Set RemEnt =.entity
                            .Remove RemEnt
                        End If
                        If (.EOF=False And .Length>0) Then
                            .MoveNext
                        End If
                    Wend
                End If
            End With

            '2.Remove this ProductID from the Collection;
            ob("Products").Collection.Remove CStr(ProductId)

            '3.Release one entity representing the Product.
            Set newEntity=ob.CloneEntity(ProEnt)
            ob.TransferOut newEntity, ob.ConnectionPoints("Out")

            Exit F[illegible]        'This function is triggered once an entity enters in "Assembly"
element. It can only cause
                        'one product to be assemblied. So it can stop
searching the remaining Products by "Exit For".
        End If
    Next

End Sub
```

KanbanSender

```
Option Explicit
Dim j As Integer

Public Function KanbanSender_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y As
Single) As Boolean
    ob.OnCreate x,y,True
    KanbanSender_OnCreate=True
    ob.AddAttribute "Name","Description",CFC_Text, CFC_Single, CFC_ReadWrite
    ob.AddAttribute "SignalName","Information To Send (An Attribute Name of The Passing
Entity)",CFC_Text, CFC_Single, CFC_ReadWrite
    ob.AddAttribute "KanbanReceiver","Signal Receiver (The Name of The
KanbanReceiver)",CFC_Text,CFC_Single, CFC_ReadWrite

    ob.AddAttribute "KanbanReceiverEle","",CFC_Numeric, CFC_Single, CFC_Hidden 'record the
KanbanReceiver element id

    ob.SetNumCoordinates 2
```

```
        ob.CoordinatesX(0)=x
        ob.CoordinatesY(0)=y
        ob.CoordinatesX(1)=x+80
        ob.CoordinatesY(1)=y+60

        ob("Name")="KanbanSender"
        ob("SignalName")="DevisionNumber"
        ob("KanbanReceiver")="KanbanReceiver1"

        ob.AddConnectionPoint "In" , x-10, y+30, CInput, 5
        ob.AddConnectionPoint "Out",x+90,y+30,COutput,5

End Function

Public Sub KanbanSender_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub KanbanSender_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.ChangeFont "Courier New",13,True,False,False,False

        CDC.RenderPicture "KanbanSender.bmp",ob.CoordinatesX(0)+1,ob.CoordinatesY(0)+1,32,32
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)
        CDC.TextOut ob.CoordinatesX(0)+5,ob.CoordinatesY(0)+40, "KanbanSender"

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)

        ob.DrawConnectionPoints

        CDC.ChangeFont "Arial",11,True,False,False,False

End Sub

Public Sub KanbanSender_OnSimulationInitialize(ob As CFCSim_ModelingElementInstance)
        Dim KanbanReceiver As CFCSim_ModelingElementInstance
                Dim i As Long

        'Locate KanbanReceiver element, which this KanbanSender wants to send singal to.
        For Each KanbanReceiver In Elements
                        Tracer.Trace "Checked " & i & " elements"     'OK!

                If KanbanReceiver.ElementType="KanbanReceiver" Then

                        If KanbanReceiver("Name")=ob("KanbanReceiver") Then
                                ob("KanbanReceiverEle")=CStr(KanbanReceiver.Id)
                                Tracer.Trace "Found KanbanReceiver"     'OK!

                                Exit For
                        End If

                End If
```

```
                        i=i+1
            Next

End Sub

Public Sub KanbanSender_OnSimulationInitializeRun(ob As CFCSim_ModelingElementInstance, RunNum
As Integer)
            j=0
End Sub

Public Sub KanbanSender_OnSimulationTransferIn(ob As CFCSim_ModelingElementInstance, Entity As
CFCSim_Entity, SrcCp As CFCSim_ConnectionPoint, DstCp As CFCSim_ConnectionPoint)
            Dim KanbanReceiver As CFCSim_ModelingElementInstance

            Set KanbanReceiver=Elements(CStr(ob("KanbanReceiverEle"))) 'The KanbanReceiver becomes
the real KanbanReceiver element existing in the model.

            'Add a record in the array Priority of KanbanReceiver element.
            KanbanReceiver("Priority").ValueRC(j,0)=Entity(CStr(ob("SignalName")))
            KanbanReceiver("Priority").ValueRC(j,1)=100000/CInt(SimTime)

            j=j+1

End Sub
```

KanbanReceiver

```
Option Explicit

Public Function KanbanReceiver_OnCreate(ob As CFCSim_ModelingElementInstance, x As Single, y As
Single) As Boolean
            ob.OnCreate x,y,True
            KanbanReceiver_OnCreate=True

            ob.AddAttribute "Name","Description",CFC_Text, CFC_Single, CFC_ReadWrite
            ob.AddAttribute "SignalName","Information To Receiver (An Attribute Name of The Passing
Entity)",CFC_Text, CFC_Single, CFC_ReadWrite
            ob.AddAttribute "ValueToUpdate","The Value to Update (An Attribute Name of The Passing
Entity)",CFC_Text,CFC_Single, CFC_ReadWrite

            ob.AddAttribute "Priority","",CFC_Array, CFC_Table, CFC_Hidden 'An internal variable to
facilitate to update attribute value of passing entities.

            ob.SetNumCoordinates 2
            ob.CoordinatesX(0)=x
            ob.CoordinatesY(0)=y
            ob.CoordinatesX(1)=x+80
            ob.CoordinatesY(1)=y+60

            ob("Name")="KanbanReceiver1"
            ob("SignalName")="DevisionNumber"
            ob("ValueToUpdate")="Priority"

            ob.AddConnectionPoint "In" , x-10, y+30, CInput, 5
```

```
        ob.AddConnectionPoint "Out",x+90,y+30,COutput,5

End Function

Public Sub KanbanReceiver_OnDragDraw(ob As CFCSim_ModelingElementInstance)
        ob.OnDraw
End Sub

Public Sub KanbanReceiver_OnDraw(ob As CFCSim_ModelingElementInstance)
        CDC.ChangeFont "Courier New",13,True,False,False,False

        CDC.RenderPicture "KanbanReceiver.bmp",ob.CoordinatesX(0)+1,ob.CoordinatesY(0)+1,32,32
        CDC.Rectangle ob.CoordinatesX(0),ob.CoordinatesY(0),ob.CoordinatesX(1),ob.CoordinatesY(1)
        CDC.TextOut ob.CoordinatesX(0)+5,ob.CoordinatesY(0)+40, "KanbanReceiver"

        If ob.Selected Then
                CDC.ChangeLineStyle CFC_DOT,1,RGB(255,0,0)
                CDC.Rectangle ob.CoordinatesX(0)-2,ob.CoordinatesY(0)-
2,ob.CoordinatesX(1)+2,ob.CoordinatesY(1)+2
        End If
        CDC.ChangeLineStyle CFC_SOLID,1,RGB(0,0,0)

        ob.DrawConnectionPoints

        CDC.ChangeFont "Arial",11,True,False,False,False

End Sub

Public Sub KanbanReceiver_OnSimulationInitialize(ob As CFCSim_ModelingElementInstance)
        ob.AddEvent "FireEntity",True

'Define Size of Array Inside Simphony
        ob("Priority").SetRC(1000,2) 'Define size of Array (Row,Column). In this array, only two: Signal
and Priority

'Give Field Names
        ob("Priority").ColumnLabel(0)= "SignalValue"
        ob("Priority").ColumnLabel(1)= "PriorityValue"

End Sub

Public Sub KanbanReceiver_OnSimulationTransferIn(ob As CFCSim_ModelingElementInstance, Entity As
CFCSim_Entity, SrcCp As CFCSim_ConnectionPoint, DstCp As CFCSim_ConnectionPoint)
        Dim j As Integer

        'Find the record and give the Priority value to the passing entity
        With ob("Priority")
                        For j=0 To 1000
                                If Entity(CStr(ob("SignalName")))=.ValueRC(j-1+1,0) Then
                                        Entity("Priority")=.ValueRC(j-1+1,1)
                                End If
                        Next j
        End With

End Sub
```

APPENDIX 3 – A COMPLETE CSOM FOR THE DOMAIN OF INDUSTRIAL CONSTRUCTION

1. XML FILE OF THE CSOM

```
<?xml version="1.0" encoding="utf-8"?>
<objectModel name="IndustrialConstruction" type="FOM">
  <objects>
    <objectClass name="HLAobjectRoot" sharing="Neither">
      <attribute name="HLAprivilegeToDeleteObject" dataType="NA"
order="TimeStamp" sharing="Publish" transportation="HLAreliable" />
      <objectClass name="Project" sharing="PublishSubscribe">
        <attribute name="Location" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
        <attribute name="Client" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
        <attribute name="EarnedValue" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
        <attribute name="ProjectName" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
        <objectClass name="Job" sharing="PublishSubscribe">
          <attribute name="JobNumber" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="MainContact" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
        </objectClass>
      </objectClass>
      <objectClass name="Product" sharing="PublishSubscribe">
        <attribute name="Description" dataType="String" order="Receive"
sharing="PublishSubscribe" transportation="HLAreliable" />
        <attribute name="ProductID" dataType="String" order="Receive"
sharing="PublishSubscribe" transportation="HLAreliable" />
        <attribute name="ProjectName" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
        <attribute name="JobNumber" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
        <objectClass name="Spool" sharing="PublishSubscribe">
          <attribute name="DevisionNumber" dataType="String"
order="Receive" sharing="PublishSubscribe" transportation="HLAreliable"
/>
          <attribute name="Thinkness" dataType="String" order="Receive"
sharing="PublishSubscribe" transportation="HLAreliable" />
          <attribute name="TotalNumSpoolsAtSameLayer" dataType="String"
order="Receive" sharing="PublishSubscribe" transportation="HLAreliable"
/>
```

```
          <attribute name="Priority" dataType="String" order="Receive"
sharing="PublishSubscribe" transportation="HLAreliable" />
          <attribute name="NeedsQC-MT" dataType="HLAboolean"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="NeedsQC-HT" dataType="HLAboolean"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="SpoolState" dataType="SpoolStateType"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="ModuleLayer" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="ModuleNumber" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="NeedsStressRelieve" dataType="HLAboolean"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="NeedsPainting" dataType="HLAboolean"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="NeedsHydroTest" dataType="HLAboolean"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="SpoolType" dataType="SpoolType"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="SpoolNumber" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="TotalDiaInch" dataType="HLAinteger"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="Length" dataType="HLAsingle"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="Weight" dataType="HLAsingle"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="NeedsCutting" dataType="HLAboolean"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="NeedsFitting" dataType="HLAboolean"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="NeedsWelding" dataType="HLAboolean"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="NeedsQC-RT" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="MaterialType" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="MainDiaInch" dataType="HLAinteger"
```

```
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="ControlNumber" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="NeedsQC-LT" dataType="HLAboolean"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
        </objectClass>
        <objectClass name="Devision" sharing="PublishSubscribe">
          <attribute name="DevisionNumber" dataType="String"
order="Receive" sharing="PublishSubscribe" transportation="HLAreliable"
/>
        </objectClass>
        <objectClass name="SteelPiece" sharing="PublishSubscribe">
          <attribute name="ModuleNumber" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
        </objectClass>
        <objectClass name="Material" sharing="PublishSubscribe">
          <objectClass name="ModuleEquipment"
sharing="PublishSubscribe">
            <attribute name="ModuleNumber" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          </objectClass>
        </objectClass>
        <objectClass name="PipeModule" sharing="PublishSubscribe">
          <attribute name="ModuleType" dataType="PipeModuleType"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="NumberOfSpoolsAtLayerA"
dataType="HLAunicodeString" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
          <attribute name="NumberOfSpoolsAtLayerB"
dataType="HLAunicodeString" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
          <attribute name="DevisionNumber" dataType="String"
order="Receive" sharing="PublishSubscribe" transportation="HLAreliable"
/>
          <attribute name="NumberOfSpoolsAtLayerC"
dataType="HLAunicodeString" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
          <attribute name="NumberOfSpoolsAtLayerD"
dataType="HLAunicodeString" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
          <attribute name="NumberOfSpoolsAtLayerE"
dataType="HLAunicodeString" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
          <attribute name="PipeModuleState"
dataType="PipeModuleStateType" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
          <attribute name="ModuleNumber" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="NumberOfLayers" dataType="HLAinteger"
order="TimeStamp" sharing="PublishSubscribe"
```

```
transportation="HLAreliable" />
        </objectClass>
      </objectClass>
      <objectClass name="HLAmanager" sharing="Neither">
        <objectClass name="HLAfederation" sharing="Publish">
          <attribute name="HLAfederationName"
dataType="HLAunicodeString" order="Receive" sharing="Publish"
transportation="HLAreliable" />
          <attribute name="HLAfederatesInFederation"
dataType="HLAfederateHandleSet" order="Receive" sharing="Publish"
transportation="HLAreliable" />
          <attribute name="HLARTIversion" dataType="HLAunicodeString"
order="Receive" sharing="Publish" transportation="HLAreliable" />
          <attribute name="HLAFDDID" dataType="HLAunicodeString"
order="Receive" sharing="Publish" transportation="HLAreliable" />
        </objectClass>
        <objectClass name="HLAfederate" sharing="Publish">
          <attribute name="HLAlookahead" dataType="HLAtimeInterval"
order="Receive" sharing="Publish" transportation="HLAreliable" />
          <attribute name="HLAGALT" dataType="HLAlogicalTime"
order="Receive" sharing="Publish" transportation="HLAreliable" />
          <attribute name="HLAROlength" dataType="HLAinteger"
order="Receive" sharing="Publish" transportation="HLAreliable" />
          <attribute name="HLATSOlength" dataType="HLAinteger"
order="Receive" sharing="Publish" transportation="HLAreliable" />
          <attribute name="HLAfederateHandle"
dataType="HLAfederateHandle" order="Receive" sharing="Publish"
transportation="HLAreliable" />
          <attribute name="HLAfederateType" dataType="HLAunicodeString"
order="Receive" sharing="Publish" transportation="HLAreliable" />
          <attribute name="HLAfederateHost" dataType="HLAunicodeString"
order="Receive" sharing="Publish" transportation="HLAreliable" />
          <attribute name="HLARTIversion" dataType="HLAunicodeString"
order="Receive" sharing="Publish" transportation="HLAreliable" />
          <attribute name="HLAFDDID" dataType="HLAunicodeString"
order="Receive" sharing="Publish" transportation="HLAreliable" />
          <attribute name="HLAtimeConstrained" dataType="HLAboolean"
order="Receive" sharing="Publish" transportation="HLAreliable" />
          <attribute name="HLAtimeRegulating" dataType="HLAboolean"
order="Receive" sharing="Publish" transportation="HLAreliable" />
          <attribute name="HLAtimeManagerState" dataType="HLAtimeState"
order="Receive" sharing="Publish" transportation="HLAreliable" />
          <attribute name="HLAlogicalTime" dataType="HLAlogicalTime"
order="Receive" sharing="Publish" transportation="HLAreliable" />
        </objectClass>
      </objectClass>
      <objectClass name="Resource" sharing="PublishSubscribe">
        <attribute name="Cost" dataType="HLAdouble" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
        <attribute name="Description" dataType="String" order="Receive"
sharing="PublishSubscribe" transportation="HLAreliable" />
        <objectClass name="Equipment" sharing="PublishSubscribe">
          <attribute name="Manufacturer" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="ID" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
```

```
transportation="HLAreliable" />
     <attribute name="LoadCapacity" dataType="HLAsingle"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
     <attribute name="ActorState" dataType="ActorStateType"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
     <attribute name="Name" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
     <objectClass name="BridgeCrane" sharing="PublishSubscribe" />
     <objectClass name="MobileCrane" sharing="PublishSubscribe">
      <attribute name="Speed" dataType="HLAunicodeString"
order="Receive" sharing="PublishSubscribe" transportation="HLAreliable"
/>
     </objectClass>
     <objectClass name="Truck" sharing="PublishSubscribe">
      <attribute name="Speed" dataType="HLAunicodeString"
order="Receive" sharing="PublishSubscribe" transportation="HLAreliable"
/>
     </objectClass>
    </objectClass>
    <objectClass name="FabricationSpace"
sharing="PublishS,bscribe">
     <attribute name="Capacity" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
     <attribute name="SpaceType" dataType="SpaceType"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
    </objectClass>
    <objectClass name="Labor" sharing="PublishSubscribe">
     <attribute name="ActorState" dataType="ActorStateType"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
     <attribute name="Rate" dataType="HLAsingle" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
     <attribute name="LaborSkill" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
     <attribute name="Productivity" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
     <objectClass name="PF-Insulator" sharing="PublishSubscribe"
/>
     <objectClass name="PF-Tracer" sharing="PublishSubscribe" />
     <objectClass name="PF-TieDownCrew" sharing="PublishSubscribe"
/>
     <objectClass name="FireProof-ConcreteCrew"
sharing="PublishSubscribe" />
     <objectClass name="FireProof-FormCrew"
sharing="PublishSubscribe" />
     <objectClass name="FireProof-MeshCrew"
sharing="PublishSubscribe" />
     <objectClass name="ScaffoldCrew" sharing="PublishSubscribe"
/>
     <objectClass name="IW-Fill-In-Crew"
```

```
sharing="PublishSubscribe" />
        <objectClass name="PF-Rigger" sharing="PublishSubscribe" />
        <objectClass name="Welder" sharing="PublishSubscribe">
          <attribute name="WelderState" dataType="WelderStateType"
order="Receive" sharing="PublishSubscribe" transportation="HLAreliable"
/>
          <attribute name="SpoolHandle"
dataType="HLAobjectInstanceHandle" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
        </objectClass>
        <objectClass name="FirstLayerWallCrew"
sharing="PublishSubscribe">
          <attribute name="WallCrewState" dataType="CutterStateType"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
        </objectClass>
        <objectClass name="QCCrew" sharing="PublishSubscribe">
          <attribute name="SpoolHandle"
dataType="HLAobjectInstanceHandle" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
          <attribute name="QCCrewState" dataType="CutterStateType"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
        </objectClass>
        <objectClass name="DraftChecker" sharing="PublishSubscribe">
          <attribute name="CheckerState" dataType="CheckerStateType"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="ShopDrawingHandle"
dataType="HLAobjectInstanceHandle" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
        </objectClass>
        <objectClass name="MaterialChecker"
sharing="PublishSubscribe">
          <attribute name="ShopDrawingHandle"
dataType="HLAobjectInstanceHandle" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
          <attribute name="MaterialCheckerState"
dataType="MaterialCheckerStateType" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
        </objectClass>
        <objectClass name="DraftPerson" sharing="PublishSubscribe">
          <attribute name="DraftPersonState"
dataType="DrafterStateType" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
          <attribute name="ShopDrawingHandle"
dataType="HLAobjectInstanceHandle" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
        </objectClass>
        <objectClass name="Cutter" sharing="PublishSubscribe">
          <attribute name="SpoolHandle"
dataType="HLAobjectInstanceHandle" order="TimeStamp"
sharing="Publis,Subscribe" transportation="HLAreliable" />
          <attribute name="CutterState" dataType="CutterStateType"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
        </objectClass>
```

```
          <objectClass name="Fitter" sharing="PublishSubscribe">
            <attribute name="SpoolHandle"
dataType="HLAobjectInstanceHandle" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
            <attribute name="FitterState" dataType="FitterStateType"
order="Receive" sharing="PublishSubscribe" transportation="HLAreliable"
/>
          </objectClass>
        </objectClass>
      </objectClass>
      <objectClass name="Information" sharing="PublishSubscribe">
        <attribute name="Description" dataType="HLAunicodeString"
order="Receive" sharing="PublishSubscribe" transportation="HLAreliable"
/>
        <attribute name="ProjectName" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
        <attribute name="JobNumber" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
        <objectClass name="Kanban" sharing="PublishSubscribe">
          <attribute name="KanbanNumber" dataType="String"
order="Receive" sharing="PublishSubscribe" transportation="HLAreliable"
/>
          <attribute name="ProductIDtoRequest" dataType="String"
order="Receive" sharing="PublishSubscribe" transportation="HLAreliable"
/>
        </objectClass>
        <objectClass name="Drawing" sharing="PublishSubscribe">
          <attribute name="Priority" dataType="HLAinteger"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <attribute name="RevisionTimes" dataType="HLAinteger"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
          <objectClass name="ShopDrawing" sharing="PublishSubscribe">
            <attribute name="ControlNumber" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
            <attribute name="ShopDrawingState"
dataType="ShopDrawingStateType" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
          </objectClass>
          <objectClass name="ISODrawing" sharing="PublishSubscribe">
            <attribute name="ISODrawingNumber"
dataType="HLAunicodeString" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
          </objectClass>
        </objectClass>
        <objectClass name="NCR" sharing="PublishSubscribe">
          <attribute name="NCRNumber" dataType="HLAunicodeString"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable" />
        </objectClass>
        <objectClass name="ChangeOrder" sharing="PublishSubscribe">
          <attribute name="ChangeOrderNumber"
dataType="HLAunicodeString" order="TimeStamp"
```

```
sharing="PublishSubscribe" transportation="HLAreliable" />
          <attribute name="ISODrawingNumber"
dataType="HLAunicodeString" order="TimeStamp"
sharing="PublishSubscribe" transportation="HLAreliable" />
        </objectClass>
      </objectClass>
    </objectClass>
  </objects>
  <interactions>
    <interactionClass name="HLAinteractionRoot" order="Receive"
sharing="Neither" transportation="HLAreliable">
      <interactionClass name="AttributeOwnershipTransfered"
order="TimeStamp" sharing="PublishSubscribe"
transportation="HLAreliable">
        <parameter name="ObjectInstance"
dataType="HLAobjectInstanceHandle" />
      </interactionClass>
      <interactionClass name="HLAmanager" order="Receive"
sharing="Neither" transportation="HLAreliable">
        <interactionClass name="HLAfederate" order="Receive"
sharing="Neither" transportation="HLAreliable">
          <parameter name="HLAfederate" dataType="HLAfederateHandle" />
          <interactionClass name="HLAreport" order="Receive"
sharing="Neither" transportation="HLAreliable">
            <interactionClass
name="HLAreportSynchronizationPointStatus" order="Receive"
sharing="Publi,h" transportation="HLAreliable">
              <parameter name="HLAsyncPointFederates"
dataType="HLAsyncPointStatusMap" />
              <parameter name="HLAsyncPointName"
dataType="HLAunicodeString" />
            </interactionClass>
            <interactionClass name="HLAreportObjectClassPublication"
order="Receive" sharing="Publish" transportation="HLAreliable">
              <parameter name="HLAnumberOfClasses"
dataType="HLAinteger" />
              <parameter name="HLAobjectClass"
dataType="HLAobjectClassHandle" />
              <parameter name="HLAattributeList"
dataType="HLAattributeHandleSet" />
            </interactionClass>
            <interactionClass name="HLAreportInteractionPublication"
order="Receive" sharing="Publish" transportation="HLAreliable">
              <parameter name="HLAinteractionClassList"
dataType="HLAinteractionClassHandleSet" />
            </interactionClass>
            <interactionClass name="HLAreportObjectClassSubscription"
order="Receive" sharing="Publish" transportation="HLAreliable">
              <parameter name="HLAnumberOfClasses"
dataType="HLAinteger" />
              <parameter name="HLAobjectClass"
dataType="HLAobjectClassHandle" />
              <parameter name="HLAactive" dataType="HLAboolean" />
              <parameter name="HLAattributeList"
dataType="HLAattributeHandleSet" />
            </interactionClass>
            <interactionClass name="HLAreportInteractionSubscription"
```

```
order="Receive" sharing="Publish" transportation="HLAreliable">
              <parameter name="HLAinteractionClassList"
dataType="HLAinteractionClassHandleSet" />
            </interactionClass>
            <interactionClass name="HLAreportSynchronizationPoints"
order="Receive" sharing="Publish" transportation="HLAreliable">
              <parameter name="HLAsyncPoints"
dataType="HLAsynchronizationPointSet" />
            </interactionClass>
          </interactionClass>
          <interactionClass name="HLArequest" order="Receive"
sharing="Neither" transportation="HLAreliable">
            <interactionClass name="HLArequestPublications"
order="Receive" sharing="Subscribe" transportation="HLAreliable" />
            <interactionClass name="HLArequestSubscriptions"
order="Receive" sharing="Subscribe" transportation="HLAreliable" />
            <interactionClass name="HLArequestSynchronizationPoints"
order="Receive" sharing="Subscribe" transportation="HLAreliable" />
            <interactionClass
name="HLArequestSynchronizationPointStatus" order="Receive"
sharing="Subscribe" transportation="HLAreliable" />
          </interactionClass>
        </interactionClass>
      </interactionClass>
    </interactionClass>
  </interactions>
  <dataTypes>
    <basicDataRepresentations>
      <basicData name="HLAinteger16LE" size="16"
interpretation="Integer in the range [-2^15, 2^15 - 1]" endian="Little"
encoding="16-bit two's complement signed integer. The most significant
bit contains the sign." />
      <basicData name="HLAinteger32LE" size="32"
interpretation="Integer in the range [-2^31, 2^31 - 1]" endian="Little"
encoding="32-bit two's complement signed integer. The most significant
bit contains the sign." />
      <basicData name="HLAinteger64LE" size="64"
interpretation="Integer in the range [-2^63, 2^63 - 1]" endian="Little"
encoding="64-bit two's complement signed integer. The most significant
bit contains the sign." />
      <basicData name="HLAfloat32LE" size="32" interpretation="Single-
precision floating point number" endian="Little" encoding="32-bit IEEE
normalized single-precision format. See IEEE Std 754-1985" />
      <basicData name="HLAfloat64LE" size="64" interpretation="Double-
precision floating point number" endian="Little" encoding="64-bit IEEE
normalized double-precision format. See IEEE Std 754-1985" />
      <basicData name="HLAoctetPairLE" size="16" interpretation="16-bit
value" endian="Little" encoding="Assumed to be portable among hardware
devices." />
      <basicData name="HLAoctet" size="8" interpretation="8-bit value"
endian="Big" e,coding="Assumed to be portable among hardware devices."
/>
    </basicDataRepresentations>
    <simpleDataTypes>
      <simpleData name="HLAattributeHandle"
representation="HLAinteger32LE" units="NA" resolution="NA"
accuracy="NA" semantics="HLA attribute handle" />
```

```
      <simpleData name="HLAbyte" representation="HLAoctet" units="NA"
resolution="NA" accuracy="NA" semantics="Uninterpreted 8-bit byte" />
      <simpleData name="HLAdouble" representation="HLAfloat64LE"
units="NA" resolution="NA" accuracy="NA" semantics="A double-precision
(64-bit) floating-point number" />
      <simpleData name="HLAfederateHandle"
representation="HLAinteger32LE" units="NA" resolution="NA"
accuracy="NA" semantics="HLA federate handle" />
      <simpleData name="HLAinteger" representation="HLAinteger32LE"
units="NA" resolution="NA" accuracy="NA" semantics="A 32-bit signed
integer" />
      <simpleData name="HLAinteractionClassHandle"
representation="HLAinteger32LE" units="NA" resolution="NA"
accuracy="NA" semantics="HLA interaction class handle" />
      <simpleData name="HLAlogicalTime" representation="HLAinteger64LE"
units="NA" resolution="NA" accuracy="NA" semantics="HLA logical time"
/>
      <simpleData name="HLAlong" representation="HLAinteger64LE"
units="NA" resolution="NA" accuracy="NA" semantics="A 64-bit signed
integer" />
      <simpleData name="HLAunicodeChar" representation="HLAoctetPairLE"
units="NA" resolution="NA" accuracy="NA" semantics="Unicode UTF-16
character (see The Unicode Standard, Version 3.0)" />
      <simpleData name="HLAobjectClassHandle"
representation="HLAinteger32LE" units="NA" resolution="NA"
accuracy="NA" semantics="HLA object class handle" />
      <simpleData name="HLAobjectInstanceHandle"
representation="HLAinteger32LE" units="NA" resolution="NA"
accuracy="NA" semantics="HLA object instance handle" />
      <simpleData name="HLAparameterHandle"
representation="HLAinteger32LE" units="NA" resolution="NA"
accuracy="NA" semantics="HLA parameter handle" />
      <simpleData name="HLAshort" representation="HLAinteger16LE"
units="NA" resolution="NA" accuracy="NA" semantics="A 16-bit signed
integer" />
      <simpleData name="HLAsingle" representation="HLAfloat32LE"
units="NA" resolution="NA" accuracy="NA" semantics="A single-precision
(32-bit) floating-point number" />
      <simpleData name="HLAtimeInterval"
representation="HLAinteger64LE" units="NA" resolution="NA"
accuracy="NA" semantics="HLA logical time interval" />
    </simpleDataTypes>
    <enumeratedDataTypes>
      <enumeratedData name="HLAboolean" representation="HLAoctet"
semantics="Standard boolean type">
        <enumerator name="HLAfalse" values="0" />
        <enumerator name="HLAtrue" values="1" />
      </enumeratedData>
      <enumeratedData name="HLAsyncPointStatus"
representation="HLAinteger32LE" semantics="Joined federate
synchronization point status">
        <enumerator name="NoActivity" values="0" />
        <enumerator name="AttemptingToRegisterSyncPoint" values="1" />
        <enumerator name="MovingToSyncPoint" values="2" />
        <enumerator name="WaitingForRestOfFederation" values="3" />
      </enumeratedData>
      <enumeratedData name="HLAtimeState"
```

```
representation="HLAinteger32LE" semantics="State of time advancement">
        <enumerator name="TimeGranted" values="0" />
        <enumerator name="TimeAdvancing" values="1" />
      </enumeratedData>
      <enumeratedData name="SpoolType" representation="HLAinteger32LE"
semantics="NA">
        <enumerator name="ModuleSpool" values="0" />
        <enumerator name="NonModuleSpool" values="1" />
      </enumeratedData>
      <enumeratedData name="ModuleLayerNumber"
representation="HLAinteger32LE" semantics="NA">
        <enumerator name="A" values="1" />
        <enumerator name="B" values="2" />
        <enumerator name="C" values="3" />
        <enumerator name="D" values="4" />
        <enumerator name="E" values="5" />
      </enumeratedData>
      <enumeratedData name="ActorStateType"
representation="HLAinteger32LE" semantics="NA",
        <enumerator name="Idle" values="0" />
        <enumerator name="Busy" values="1" />
      </enumeratedData>
      <enumeratedData name="SpaceType" representation="HLAinteger32LE"
semantics="NA">
        <enumerator name="Shop" values="0" />
        <enumerator name="WorkCell" values="1" />
        <enumerator name="LayDown" values="2" />
        <enumerator name="Yard" values="3" />
      </enumeratedData>
      <enumeratedData name="LayborType" representation="HLAinteger32LE"
semantics="NA">
        <enumerator name="Drafter" values="0" />
        <enumerator name="Checker" values="1" />
        <enumerator name="Cutter" values="2" />
        <enumerator name="Fitter" values="3" />
        <enumerator name="Welder" values="4" />
        <enumerator name="QC" values="5" />
      </enumeratedData>
      <enumeratedData name="LaborSkillType"
representation="HLAinteger32LE" semantics="NA">
        <enumerator name="Low" values="0" />
        <enumerator name="Medium" values="1" />
        <enumerator name="High" values="2" />
      </enumeratedData>
      <enumeratedData name="SpoolStateType"
representation="HLAinteger32LE" semantics="NA">
        <enumerator name="Cutted" values="1" />
        <enumerator name="Welded" values="2" />
        <enumerator name="QCChecked" values="3" />
        <enumerator name="HydroTested" values="5" />
        <enumerator name="StressRelieved" values="4" />
        <enumerator name="Painted" values="6" />
        <enumerator name="Drafted" values="0" />
      </enumeratedData>
      <enumeratedData name="ShopDrawingStateType"
representation="HLAinteger32LE" semantics="NA">
        <enumerator name="Drafted" values="1" />
```

```
            <enumerator name="Checked" values="2" />
            <enumerator name="ReDrafted" values="3" />
            <enumerator name="ReChecked" values="4" />
            <enumerator name="MaterialChecked" values="5" />
            <enumerator name="BeingFabricated" values="7" />
            <enumerator name="Undrafted" values="0" />
            <enumerator name="MaterialBeingProcured" values="6" />
         </enumeratedData>
         <enumeratedData name="PipeModuleStateType"
representation="HLAinteger32LE" semantics="NA">
            <enumerator name="FirstLayerFinished" values="1" />
            <enumerator name="SecondLayerFinished" values="2" />
            <enumerator name="ThirdLayerFinished" values="3" />
            <enumerator name="FourthLayerFinished" values="4" />
            <enumerator name="Fabricated" values="0" />
         </enumeratedData>
         <enumeratedData name="DraftPersonStateType"
representation="HLAinteger32LE" semantics="NA" />
         <enumeratedData name="CheckerStateType"
representation="HLAinteger32LE" semantics="NA" />
         <enumeratedData name="CutterStateType"
representation="HLAinteger32LE" semantics="NA">
            <enumerator name="FitterStateType" values="0" />
         </enumeratedData>
         <enumeratedData name="WelderStateType"
representation="HLAinteger32LE" semantics="NA" />
         <enumeratedData name="FitterStateType"
representation="HLAinteger32LE" semantics="NA" />
         <enumeratedData name="MaterialCheckerStateType"
representation="HLAinteger32LE" semantics="NA" />
         <enumeratedData name="PipeModuleType"
representation="HLAinteger32LE" semantics="NA">
            <enumerator name="Equipment" values="0" />
            <enumerator name="Piperack" values="1" />
            <enumerator name="ElectricalTrayOnly" values="2" />
            <enumerator name="Piperack/ElectricalTray" values="3" />
         </enumeratedData>
      </enumeratedDataTypes>
      <arrayDataTypes>
         <arrayData name="HLAattributeHandleSet"
dataType="HLAattributeHandle" cardinality="Dynamic"
encoding="HLAvariableArray" semantics="Collection of attribute handles"
/>
         <arrayData name="HLAfederateHandleSet"
dataType="HLAfederateHandle" cardinality="Dynamic"
encoding="HLAvariableArray" semantics="Co[illegible]ection of federate handles"
/>
         <arrayData name="HLAinteractionClass[illegible]eSet"
dataType="HLAinteractionClassHandle" cardinality="Dynamic"
encoding=",LAvariableArray" semantics="Collection of interaction class
handles" />
         <arrayData name="HLAsynchronizationPointSet"
dataType="HLAunicodeString" cardinality="Dynamic"
encoding="HLAvariableArray" semantics="Collection of synchronization
point labels" />
         <arrayData name="HLAsyncPointStatusMap"
dataType="HLAsyncPointStatus" cardinality="Dynamic"
```

```
encoding="HLAvariableArray" semantics="Collection of federate
synchronization point statuses" />
      <arrayData name="HLAunicodeString" dataType="HLAunicodeChar"
cardinality="Dynamic" encoding="HLAvariableArray" semantics="Unicode
string representation" />
    </arrayDataTypes>
    <fixedRecordDataTypes />
  </dataTypes>
</objectModel>
```